UNIVERSITÉ LUMIÈRE LYON 2
UNIVERSITÉ DE LYON

ERIC
ENTREPÔTS, REPRÉSENTATION & INGÉNIERIE des CONNAISSANCES

Thèse

présentée pour obtenir le grade de docteur

de l'Université Lumière Lyon2

Spécialité : Informatique

présentée par

Rym KHEMIRI

Vers l'OLAP collaboratif pour la recommandation des analyses en ligne personnalisées

préparée au sein du laboratoire ERIC sous la direction de

Fadila BENTAYEB

Jérôme DARMONT

Jalel AKAICHI

à soutenir le 23 Septembre 2015 devant le jury composé de

Esteban Zimanyi	Président
Ladjel Bellatreche	Examinateur
Gilles Zurfluh	Rapporteur
Imed Riadh Farah	Rapporteur
Fadila Bentayeb	Directrice de thèse
Jérôme Darmont	Directeur de thèse
Jalel Akaichi	Directeur de thèse

Résumé

La personnalisation vise à recueillir les intérêts, les préférences, les usages, les contraintes, le contexte, etc. souvent considérés comme faisant partie de ce que l'on appelle "profil utilisateur" pour ensuite les intégrer dans un système et les exploiter afin de permettre à l'utilisateur d'accéder rapidement aux informations les plus pertinentes pour lui. Par ailleurs, au sein d'une organisation, différents acteurs sont amenés à prendre des décisions à différents niveaux de responsabilité et ont donc besoin de réaliser des analyses à partir de l'entrepôt de données pour supporter la prise de décision. Ainsi, dans le contexte de cette communauté d'utilisateurs de l'entrepôt de données, la notion de collaboration émerge. Il est alors intéressant de combiner les concepts de personnalisation et de collaboration pour approcher au mieux les besoins des utilisateurs en leur recommandant des analyses en ligne pertinentes.

L'objectif de ce mémoire est de proposer une approche collaborative pour l'OLAP, impliquant plusieurs utilisateurs, dirigée par un processus de personnalisation intégré aux systèmes décisionnels afin de pouvoir aider l'utilisateur final dans son processus d'analyse en ligne. Qu'il s'agisse de personnalisation du modèle d'entrepôt, de recommandation de requêtes décisionnelles ou de recommandation de chemins de navigation au sein des cubes de données, l'utilisateur a besoin d'un système décisionnel efficace qui l'aide dans sa démarche d'analyse en ligne. La finalité est de fournir à l'utilisateur des réponses pertinentes proches de ses besoins pour qu'il puisse mieux appréhender ses prises de décision. Nous nous sommes intéressés dans cette thèse à trois problèmes relevant de la prise en compte de l'utilisateur au sein des entrepôts de données et de l'OLAP. Nos contributions s'appuient sur la combinaison de techniques issues de la fouille de données avec les entrepôts et OLAP.

Notre première contribution est une approche qui consiste à personnaliser les hiérarchies de dimensions afin d'obtenir des axes d'analyse nouveaux sémantiquement plus riches pouvant aider l'utilisateur à réaliser de nouvelles analyses non prévues par le modèle de l'entrepôt initial. En effet, nous relâchons la contrainte du modèle fixe de l'entrepôt, ce qui permet à l'utilisateur de créer de nouveaux axes d'analyse pertinents en tenant compte à la fois de ses contraintes et des connaissances enfouies dans les données entreposées.

Notre approche repose sur une méthode d'apprentissage non-supervisé, le k-means contraint, capable de créer de nouveaux regroupements intéressants des données entreposées pouvant constituer un nouveau niveau de hiérarchie permettant de réaliser de nouvelles requêtes décisionnelles. L'intérêt est alors de pouvoir exploiter ces nouveaux niveaux de hiérarchie pour que les autres utilisateurs appartenant à la même communauté d'utilisateurs puissent en tirer profit, dans l'esprit d'un système collaboratif dans lequel chacun apporte sa pierre à l'édifice.

Notre deuxième contribution est une approche interactive pour aider l'utilisateur à formuler de nouvelles requêtes décisionnelles pour construire des cubes OLAP pertinents en s'appuyant sur ses requêtes décisionnelles passées, ce qui lui permet d'anticiper sur ses besoins d'analyse futurs. Cette approche repose sur l'extraction des motifs fréquents à partir d'une charge de requêtes associée à un ou à un ensemble d'utilisateurs appartenant à la même communauté d'acteurs d'une organisation. Notre intuition est que la pertinence d'une requête décisionnelle est fortement corrélée avec la fréquence d'utilisation par l'utilisateur (ou un ensemble d'utilisateurs) des attributs associés à l'ensemble de ses (leurs) requêtes précédentes. Notre approche de formulation de requêtes décisionnelles est collaborative puisqu'elle permet à l'utilisateur de soumettre à l'entrepôt de données des requêtes pertinentes construites, pas à pas, à partir des attributs les plus fréquemment utilisés par l'ensemble des acteurs de la communauté d'utilisateurs à laquelle il appartient.

Enfin, notre troisième contribution est une approche de recommandation des chemins de navigation au sein des cubes OLAP. Les utilisateurs sont souvent livrés à eux mêmes et ne sont pas guidés dans leur processus de navigation. Pour pallier ce problème, nous développons une approche centrée utilisateur qui suggère à l'utilisateur des conseils de navigation. En

effet, nous guidons l'utilisateur à aller vers les faits les plus intéressants dans les cubes OLAP en lui recommandant les chemins de navigation les plus pertinents pour lui. Cette approche s'appuie sur les chaînes de Markov qui permettent de prédire la prochaine requête d'analyse à partir de la seule requête courante. Ce travail s'inscrit dans une approche de recommandation collaborative car les probabilités de passage d'une requête à l'autre dans le treillis de cuboïdes (cube OLAP) est calculé en tenant compte des toutes les requêtes d'analyse de l'ensemble des utilisateurs appartenant à la même communauté.

Afin de valider nos propositions, nous présentons un système d'aide à la décision collaboratif qui se décline en deux sous-systèmes : (1) personnalisation du contenu de l'entrepôt et (2) recommandation de requêtes et de chemins de navigation dans les cubes OLAP. Nous avons mené également des expérimentations qui ont montré l'efficacité de nos approches d'analyse en ligne centrées utilisateur en utilisant des mesures de qualité telles que le rappel et la précision. En effet, les tests que nous avons effectués ont démontré que les deux fonctionnalités de notre système à savoir la personnalisation et la recommandation donnent de résultats plutôt satisfaisants.

Mots clés

Entrepôts de données, OLAP collaboratif, profil utilisateur, personnalisation, recommandation.

Abstract

Personalization is to gather the interests, preferences, uses, constraints, context, etc. often considered part of the so-called " user profile " and then integrate them into a system and operate to allow the user to quickly access the most relevant information for him.

The objective of this thesis is to provide a collaborative approach to the OLAP involving several users, led by an integrated personalization process in decision-making systems in order to help the end user in their analysis process. Whether personalizing the warehouse model, recommending decision queries or recommending navigation paths within the data cubes, the user need an efficient decision-making system that assist him. We were interested in three issues falling within data warehouse and OLAP personalization offering three major contributions. Our contributions are based on a combination of datamining techniques with data warehouses and OLAP technology.

Our first contribution is an approach about personalizing dimension hierarchies to obtain new analytical axes semantically richer for the user that can help him to realize new analyzes not provided by the original data warehouse model. Indeed, we relax the constraint of the fixed model of the data warehouse which allows the user to create new relevant analysis axes taking into account both his/her constraints and his/her requirements. Our approach is based on an unsupervised learning method, the constrained k-means. Our goal is then to recommend these new hierarchy levels to other users of the same user community, in the spirit of a collaborative system in which each individual brings his contribution.

The second contribution is an interactive approach to help the user to formulate new decision queries to build relevant OLAP cubes based on its past decision queries, allowing it to anticipate its future analysis needs. This approach is based on the extraction of frequent itemsets from a query load associated with one or a set of users belonging to the same actors in a community organization. Our intuition is that the relevance of a decision query is strongly correlated to the usage frequency of the corresponding attributes within a given workload of a user (or group of users). Indeed, our approach of decision queries formulation is a collaborative approach because it allows the user to formulate relevant queries, step by step, from the most commonly used attributes by all actors of the user community.

Our third contribution is a navigation paths recommendation approach within OLAP cubes. Users are often left to themselves and are not guided in their navigation process. To overcome this problem, we develop a user-centered approach that suggests the user navigation guidance. Indeed, we guide the user to go to the most interesting facts in OLAP cubes telling him the most relevant navigation paths for him. This approach is based on Markov chains that predict the next analysis query from the only current query. This work is part of a collaborative approach because transition probabilities from one query to another in the cuboids lattice (OLAP cube) is calculated by taking into account all analysis queries of all users belonging to the same community.

To validate our proposals, we present a support system user-centered decision which comes in two subsystems : (1) content personalization and (2) recommendation of decision queries and navigation paths. We also conducted experiments that showed the effectiveness of our analysis online user centered approaches using quality measures such as recall and precision.

Keywords

Data warehouse, collaborative OLAP, user profile, personalization, recommendation.

Table des matières

Table des figures

Liste des tableaux

Liste des Algorithmes

Chapitre 1

Introduction générale

Cette thèse s'inscrit dans le domaine des entrepôts de données et traite en particulier la prise en compte de l'utilisateur dans les entrepôts de données et l'analyse en ligne (OLAP). Ce chapitre introductif présente le contexte de nos travaux, motive notre sujet de thèse et en présente les objectifs. Tout d'abord, avant de présenter les problèmes et les enjeux liés à la prise en compte de l'utilisateur dans les entrepôts de données, nous évoquons certaines généralités sur ces derniers afin d'éclairer nos propos futurs. Il s'agit en effet de revenir sur les concepts clés des entrepôts de données et de l'analyse en ligne. Ensuite, nous motivons la prise en compte de l'utilisateur dans le processus d'analyse en ligne, nous posons les problématiques auxquelles nous nous sommes intéressés et nous exposons nos contributions. Enfin, nous présentons l'organisation de ce manuscrit.

1.1 Contexte de nos travaux

Les entrepôts de données sont apparus dans les années 90 [CCS93] pour répondre au défi de l'intégration d'une grande quantité de données, relatives à un certain domaine d'application, et stockées physiquement dans différentes sources de données hétérogènes. Ils permettent permettent de consolider, stocker et organiser ces données à des fins d'analyse.

Par conséquent, l'avènement des entrepôts de données apporte au processus décisionnel une réponse au problème de la croissance continuelle de données variées, ce qui permet de supporter efficacement l'analyse en ligne [CD97]. Ils permettent de définir une architecture qui sert de fondation aux applications décisionnelles.

La question qui se pose alors est : comment peut-on définir un entrepôt de données ? Il existe de nombreuses définitions de l'entrepôt de données, dont voici une petite sélection. Selon Inmon (souvent désigné comme le "père des entrepôts de données"), 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]. Kimball, qui est probablement le plus connu après Inmon dans le domaine des entrepôts de données, définit un entrepôt de données comme "un exemplaire de données relatives à des transactions structurées spécifiquement à des fins de consultation et d'analyse" [KR02]. Selon Widom, l'entrepôt de données est dé-

fini comme un ensemble de vues matérialisées [Wid95]. Devlin décrit quant à lui un entrepôt de données comme étant "une mémoire unique, complète et cohérente de données provenant de sources diverses et mises à la disposition des utilisateurs finaux sous une forme compréhensible et utilisable dans un contexte commercial" [Dev96]. Et enfin, selon Kelly, un entrepôt de données est "une architecture d'entreprise permettant l'exploitation des données au niveau global, dotée de normes, de principes et d'une infrastructure qui sert de base à toutes les applications d'aide à la décision" [Kel97].

Aucune de ces définitions ne suffit pour expliquer l'expression "entrepôt de données". Cependant, elles contiennent toutes des caractéristiques essentielles. Même si les théoriciens sont d'accord sur la plupart de ces caractéristiques, leurs points de vue peuvent toutefois diverger considérablement dans les détails. Ainsi, l'entrepôt de données selon Inmon ne correspond pas en tous points à celui défini par Kimball ni à celui défini par Widom, par exemple.

La définition d'un entrepôt de données selon Inmon, reconnue actuellement par la communauté scientifique, est la plus proche de la réalité (celle vécue au sein des entreprises) pour l'élaboration des systèmes d'information décisionnels (SID), puisqu'elle regroupe à la fois les processus d'intégration, de modélisation et de stockage des données pour des fins d'analyse.

Un entrepôt de données présente une modélisation dite multidimensionnelle qui consiste à définir un sujet d'analyse comme un ensemble de faits représentés dans un espace multidimensionnel [ASS01]. Les concepts de base utilisés pour construire un espace multidimensionnel sont donc les faits et les dimensions qui représentent les axes d'observation. Un fait est défini comme étant un sujet d'intérêt pour une entreprise, représentant ainsi l'objet d'analyse. Il est décrit par un ensemble d'attributs appelés indicateurs ou mesures [GMR98]. Les dimensions représentent les différentes vues possibles à partir desquelles les mesures sont observées et analysées. Les faits et les dimensions sont connus comme des données quantitatives et qualitatives respectivement. À partir de l'entrepôt de données, des contextes d'analyse sont alors extraits, représentés et visualisés à travers la métaphore du cube [CD97]. Les dimensions agissent alors en tant que coordonnées du cube et les différentes combinaisons des dimensions définissent les cellules du cube qui contiennent les valeurs agrégées de la mesure. Si on a plus de trois dimensions, on parle alors d'hypercube.

La construction d'un entrepôt de données constitue une phase essentielle dans le processus d'entreposage et consomme, généralement, jusqu'à 80 % du temps de développement d'un entrepôt de données. C'est la procédure de migration des données de production dans le système décisionnel après leurs homogénéisation via des opérations de sélection, de nettoyage et de reformatage. Cette phase est connue sous le nom d'ETL (Extract-Transform-Load), puisqu'elle permet d'extraire, de transformer et puis de charger les données des sources vers l'entrepôt.

Ensuite, c'est la phase d'analyse en ligne qui prend lieu, souvent désignée par analyse multidimensionnelle, en référence au type des données manipulées, ou par analyse OLAP, en référence à la technologie utilisée (Figure 1.1). Dans un premier temps, l'utilisateur formule la requête décisionnelle qui permet de produire le cube de données en fonction de ses besoins d'analyse. Cela correspond à la notion de vue matérialisée selon Widom. Dans un second temps, l'utilisateur navigue, explore et

analyse les données du cube OLAP pour en extraire des informations pertinentes pour l'aide à la prise de décision. Le cube de données est simple à manipuler et à explorer grâce aux opérateurs OLAP. Ces derniers peuvent être classés en trois catégories : les opérateurs liés à la structure (rotate, switch, etc.), les opérateurs liés à la granularité (roll-up, drill-down, etc.) et enfin les opérateurs ensemblistes (slice & dice).

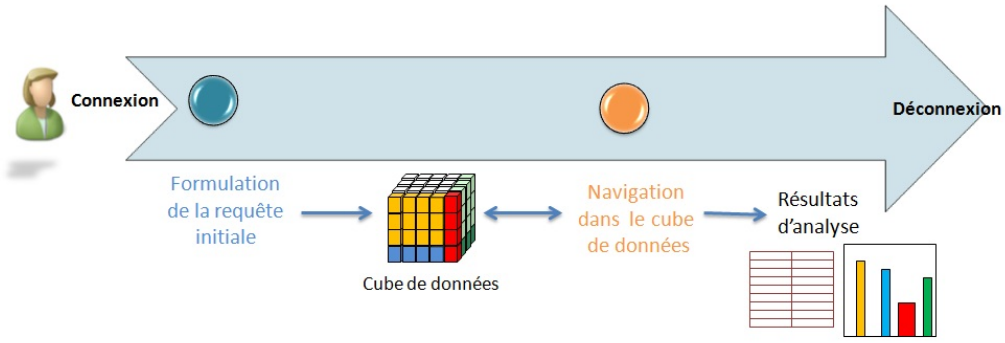


FIGURE 1.1 – Session d'analyse

De ce qui précède, nous constatons que l'interaction entre le SID et l'utilisateur est primordiale dans un processus d'aide à la décision. Une plus grande interaction de l'utilisateur avec le SID permettrait d'envisager des bénéfices tant au niveau système qu'au niveau utilisateur. En effet, la connaissance accrue de l'utilisateur par le système doit pouvoir servir à mieux paramétrer celui-ci en fonction des éléments appris. Par conséquent, un système qui s'adapte aux utilisateurs doit leur permettre une meilleure utilisation de ses fonctionnalités et une réduction de leurs efforts pour accéder et manipuler les informations pour des fins d'analyse. Cela permettrait une meilleure exploitation de l'entrepôt de données. Ainsi, l'interaction entre le décideur et le SID est un aspect crucial au niveau de la conception, la construction et l'exploitation (analyse) des entrepôts de données. Cela nécessite de s'appuyer sur une démarche centrée utilisateur dans toutes les phases du processus décisionnel.

La définition actuelle d'un entrepôt de données n'est donc pas totalement satisfaisante, notamment dans la perspective de prendre en compte l'utilisateur. En effet, la construction d'un entrepôt de données est basée uniquement sur, d'une part, les objectifs d'analyse globaux et, d'autre part, sur les sources de données. Les besoins individuels des utilisateurs sont souvent laissés de côté car ils apparaissent dans la majorité des cas pendant la phase d'exploitation ; autrement dit lorsque l'entrepôt est déjà construit.
Par ailleurs, nous constatons qu'il existe deux étapes primordiales pendant la session d'analyse d'un utilisateur. La première étape consiste à formuler la première requête décisionnelle pour la production du cube OLAP initial en fonction des besoins exprimés par l'utilisateur. La deuxième étape consiste pour l'utilisateur à entamer la

phase de navigation et d'exploration du cube OLAP qui consiste en une interrogation interactive des données, en réalisant des analyses à travers de multiples passes (passant par exemple des données résumées à des données détaillées ou changeant même d'axe d'observation), successivement dans des niveaux de détail inférieurs permettant ainsi de produire les premiers résultats d'analyse pertinents pour l'aide à la décision.

Partant de ce constat, il apparaît évident que le rôle de l'utilisateur est ici central puisque c'est à lui d'analyser et d'explorer le cube de données. Dans ce contexte, les systèmes OLAP doivent intégrer l'utilisateur dans le processus d'analyse en ligne lui permettant ainsi une meilleure prise en main de l'outil décisionnel qui pourra lui fournir des informations se rapprochant le plus possible de ses besoins analytiques et de ses intérêts individuels.

Dans cette section, nous avons évoqué les principaux concepts liés à la modélisation et à l'exploitation des entrepôts de données. Cette présentation nous a permis de positionner le contexte général de nos travaux et de constater en particulier que le rôle de l'utilisateur est central dans un système OLAP. C'est dans ce contexte que nous nous sommes intéressés à la problématique de la prise en compte de l'utilisateur dans les entrepôts de données et au cœur du processus d'analyse en ligne. Ainsi, nous proposons d'aborder dans la section suivante les problématiques de recherche liées aux entrepôts de données et à l'analyse en ligne selon la perspective utilisateur. Nous présentons notamment certaines limites des entrepôts de données "classiques" relatives à la prise en compte de l'utilisateur que nous avons identifiées et auxquelles nous avons tenté d'apporter des solutions adaptées.

1.2 Problématiques

Un entrepôt de données est conçu pour répondre à un ensemble de besoins d'analyse globaux recensés auprès des utilisateurs pendant la phase de conception. Cependant, une fois l'entrepôt de données construit, de nouveaux besoins des utilisateurs peuvent émerger. En effet, chaque utilisateur, lors de la phase d'exploitation de l'entrepôt de données, peut exprimer des besoins spécifiques en fonction de ses propres intérêts, usages, caractéristiques, contraintes, contextes et préférences auxquels l'entrepôt n'est pas forcément en mesure d'apporter des réponses. Il est donc difficile de recenser de façon exhaustive les besoins et caractéristiques de tous les utilisateurs, d'autant que ceux-ci peuvent évoluer dans le temps. Dans ce contexte, le recueil d'informations concernant l'utilisateur pour mieux prendre en compte ses caractéristiques afin de les intégrer dans le processus décisionnel devient un enjeu majeur. Cela nécessite d'impliquer davantage l'utilisateur dans toutes les phases du processus d'entreposage de données. De ce fait, une problématique nouvelle dans les entrepôts de données a émergé ces dernières années, connue sous le nom de *personnalisation*, qui pose plusieurs enjeux.
La personnalisation vise à recueillir les intérêts, les préférences, les usages, les contraintes, le contexte, etc. souvent considérés comme faisant partie de ce que l'on appelle profil utilisateur afin de les exploiter dans un système d'information pour permettre d'ac-

céder rapidement aux informations les plus pertinentes. C'est ainsi que, ces dernières années, la personnalisation dans les bases de données ou les entrepôts de données a fait l'objet de nombreuses recherches [KI04b, SP08, BFB08, JRTZ09a, GMNS09, GR09], sans pour autant avoir reçu à l'heure actuelle de solution générale.

Par ailleurs, selon les fondements définis par Codd et al. [CCS93], l'analyse en ligne repose sur une manipulation intuitive des données. Elle permet d'interroger les données de l'entrepôt par des requêtes complexes et de naviguer dans les cubes de données en utilisant les opérateurs OLAP. Cependant, l'analyse en ligne "classique " peut soulever trois problèmes majeurs.

Le premier problème concerne le schéma de l'entrepôt de données. En effet, l'OLAP ne dispose pas d'outils pour guider l'utilisateur vers de nouvelles explorations non définies par le modèle de l'entrepôt ni pour approfondir l'analyse vers, par exemple, la structuration, l'explication ou la prédiction. En effet, le processus décisionnel repose sur un modèle d'entrepôt de données fixe. Une fois l'entrepôt créé, les utilisateurs peuvent uniquement réaliser les analyses prévues par ce modèle. Cependant, un utilisateur peut vouloir anticiper la réalisation d'événements futurs en s'appuyant sur ses usages, ses sessions d'analyse passées, en y intégrant par exemple ses propres contraintes. De ce fait, un entrepôt de données avec un schéma fixé a priori n'est plus nécessairement pertinent tant qu'il ne pourra pas prendre en compte les nouveaux besoins d'analyse exprimés par l'utilisateur.

Le deuxième problème concerne la formulation de requêtes décisionnelles initiales, autrement dit, la définition du schéma du cube OLAP. Ce problème est d'autant plus crucial que l'utilisateur, ne sachant pas a priori ce qu'il cherche, pose plusieurs requêtes pour trouver ce qui l'intéresse. Il est généralement confronté à un espace multidimensionnel très vaste. Ainsi, cette étape de formulation de requêtes décisionnelles nécessite un effort non négligeable de la part de l'utilisateur. Ce problème coïncide avec le problème du choix des cubes de données à construire (matérialisation de vues) où l'utilisateur est censé sélectionner les faits à analyser définis par les mesures (indicateurs), ainsi que les les axes d'analyse et leurs niveaux de granularité.

Le troisième problème concerne la navigation dans le cube OLAP lui-même. En effet, une fois le résultat de la requête décisionnelle obtenu (cube de données), l'utilisateur est livré à lui même pendant la phase de navigation. C'est à lui en effet de manipuler au mieux le cube de données, de choisir les meilleurs chemins de navigation, pour y découvrir des zones intéressantes. Cependant, l'OLAP ne dispose pas d'outils pour guider l'utilisateur vers les faits les plus pertinents. C'est aussi à l'utilisateur d'évaluer la pertinence des informations découvertes pour savoir si elles constituent des informations répondant au mieux à ses besoins.

Enfin, tous ces problèmes ne peuvent pas être résolus facilement puisque la technologie OLAP présente des limites. Il est vrai que l'analyse OLAP offre des possibilités pour visualiser des faits décrits par des indicateurs et des axes d'analyse, mais elle ne permet pas de décrire l'ordre d'importance ou les relations possibles entre ces faits. L'OLAP ne permet pas non plus de classifier ou de regrouper les faits selon un ordre de proximité sémantique et ne dispose pas non plus de moyens pour expliquer les associations ou les implications entre ces faits. C'est ainsi que ces problèmes réduisent considérablement les avantages de l'analyse en ligne.

Par ailleurs, au sein d'une organisation, différents acteurs sont amenés à prendre

des décisions à différents niveaux de responsabilité et ont donc besoin de réaliser des analyses à partir de l'entrepôt de données pour supporter la prise de décision. Ainsi, dans le contexte de cette communauté d'utilisateurs de l'entrepôt de données, la notion de collaboration émerge. Ainsi, à l'image du Web, qui est un lieu d'informations et d'échanges qualifié de social, participatif et collaboratif, les applications décisionnelles se doivent de fournir une analyse en ligne pouvant être partagée par plusieurs utilisateurs selon leurs profils, leurs usages et plus particulièrement leurs analyses. Les utilisateurs peuvent alors tirer profit non seulement de leurs propres expériences passées mais également de celles des autres utilisateurs pour avoir des analyses partagées. Dans ce contexte, nous nous intéressons particulièrement à l'analyse en ligne collaborative. Plusieurs verrous scientifiques sont alors soulevés dans le but de faciliter le processus de personnalisation et de recommandation d'analyses en ligne collaborative :

1. Comment enrichir les modèles d'entrepôt de données par, d'une part des connaissances utilisateurs (profils, contraintes, etc.), et d'autre part par l'extraction de connaissances à partir des données entreposées afin d'anticiper la réalisation d'évènements futurs ?
2. Comment aider l'utilisateur à formuler des requêtes décisionnelles pertinentes à partir de ses propres requêtes précédentes ainsi que celles des autres utilisateurs ?
3. Comment guider l'utilisateur vers les faits les plus pertinents dans un cube OLAP à partir des sessions d'analyse précédentes de l'ensemble des utilisateurs ?

Pour chacune des questions énoncées ci-dessus, nous tentons d'apporter des solutions originales en mettant l'utilisateur au cœur du système décisionnel. Dans la section suivante, nous présentons nos contributions tout en motivant les choix qui sont les nôtres.

1.3 Objectifs et contributions

Nous avons identifié dans la section précédente plusieurs problèmes relatifs à la prise en compte de l'utilisateur dans le processus d'analyse en ligne, auxquels nous avons tenté d'apporter des solutions adaptées. Dans leur finalité, nos contributions s'inscrivent dans le cadre des travaux sur la personnalisation et la recommandation dans les entrepôts et l'analyse en ligne de données, pour développer des systèmes d'aide à la décision centrés utilisateurs. En effet, par rapport au volume de plus en plus important de données, l'accès à l'information pertinente devient un enjeu critique pour l'utilisateur final. Mais au delà de cette difficulté, il est également important pour l'utilisateur de sentir que le système ait été fait pour lui et qu'il lui répond de façon plus personnalisée. Pour atteindre cet objectif, une solution consiste à intégrer son profil dans le système décisionnel à tous les niveaux du processus d'entreposage de données.
Les contributions exposées dans ce mémoire combinent les fonctionnalités de personnalisation et de recommandation collaborative. L'intérêt est alors de pouvoir ex-

ploiter les usages d'un utilisateur donné, pour que les autres utilisateurs similaires puissent en tirer profit, dans l'esprit d'un système collaboratif. En d'autres termes, il s'agit de donner la possibilité aux utilisateurs d'un système décisionnel de partager des analyses OLAP. Habituellement, le filtrage collaboratif se base sur l'idée que des personnes similaires selon un certain nombre de critères peuvent partager des informations et/ou des résultats. L'idée d'exploiter ce que les autres utilisateurs ont fait pour produire des recommandations est très populaire dans le domaine de la recherche d'information [AT05] et dans l'exploitation des usages du Web (Web Usage Mining) [SCDT00]. En nous inspirant des idées développées dans ces travaux, nous adaptons des techniques issues de la fouille de données et de la recherche d'information afin de réaliser des recommandations collaboratives dans les systèmes d'information décisionnels. Notre objectif est de faire évoluer l'OLAP vers d'autres possibilités d'analyse plus proches des besoins réels de l'utilisateur en intégrant l'aspect collaboratif.

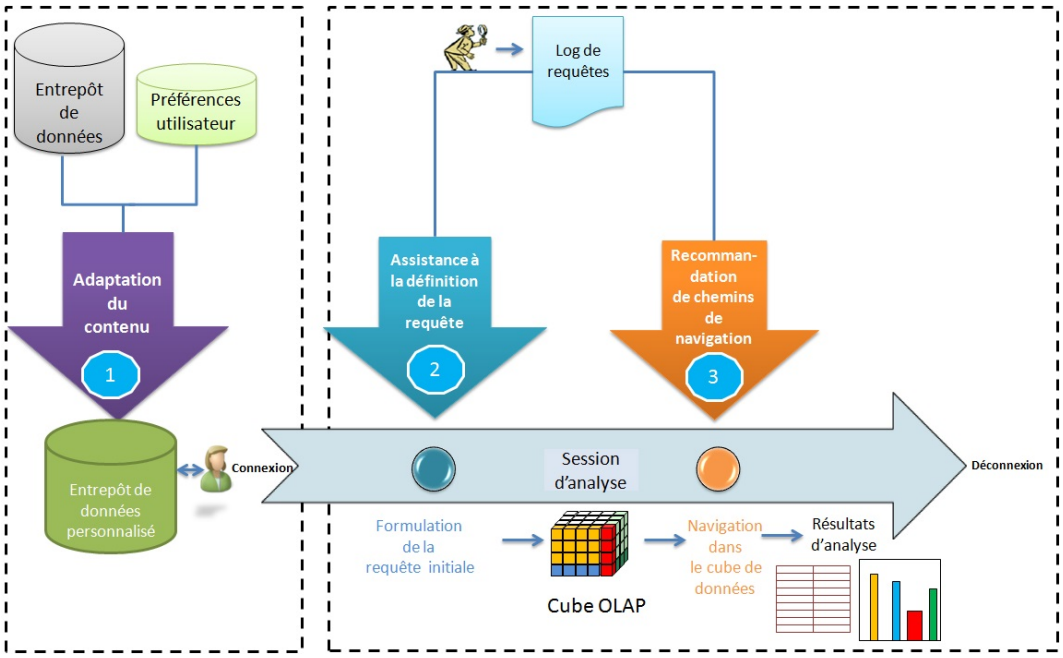


FIGURE 1.2 – Processus de personnalisation et de recommandation dans les entrepôts de données

Nous présentons dans la Figure 1.2 notre approche générale pour l'OLAP personnalisé et collaboratif. Nous présentons trois contributions. Chacune d'entre elles est liée à un événement particulier du processus d'analyse en ligne. Notre première contribution concerne la personnalisation du contenu de l'entrepôt de données par adaptation en intégrant les contraintes utilisateur. Notre deuxième contribution est directement liée au choix des requêtes décisionnelles définissant les cubes OLAP. Il s'agit d'aider l'utilisateur à formuler une nouvelle requête décisionnelle en s'appuyant sur l'historique des requêtes de l'ensemble des utilisateurs. Enfin, notre troisième contribution a pour objectif de guider l'utilisateur vers les faits les plus pertinents

lors de sa navigation dans un cube OLAP.

1.3.1 Personnalisation de dimensions dans un entrepôt de données

Notre première contribution s'inscrit dans les travaux relatifs à la personnalisation du contenu d'un entrepôt de données. D'autre part, il est important de prendre en compte les préférences utilisateur ou plus précisément les contraintes utilisateur. Il s'agit alors de faire évoluer les possibilités d'analyse de l'entrepôt de données en fonction des besoins réels des utilisateurs tout en l'adaptant à leurs contraintes.

Pour atteindre cet objectif, nous proposons une approche de personnalisation des tables de dimension d'un entrepôt de données en empruntant le concept d'évolution de schéma. L'évolution de schéma permet de faire évoluer le modèle d'entrepôt, offrant ainsi à l'utilisateur de nouvelles possibilités d'analyse.

Pour obtenir ces nouvelles possibilités d'analyse, nous utilisons la fouille de données pour extraire de nouvelles structures à partir des données entreposées contenues dans les tables de dimensions, tout en tenant compte des contraintes utilisateurs afin de les matérialiser sous forme de nouveaux niveaux de hiérarchies de dimensions.

Nous avons conçu et développé pour cela un opérateur personnalisé appelé PRoCK (Personalized Roll-up with Constrained Kmeans). PRoCK propose de bénéficier de techniques de fouille de données, plus précisément du clustering contraint, pour découvrir de nouveaux regroupements de données tout en respectant les contraintes utilisateur. En effet, appliqué sur les instances d'un niveau d'une hiérarchie de dimension donnée, le clustering contraint donne une nouvelle classification naturelle des données pouvant constituer un nouveau niveau de hiérarchie de dimension. Ce nouveau niveau de hiérarchie est sémantiquement plus riche, personnalisé et l'exploitant l'utilisateur peut élaborer des analyses OLAP plus sophistiquées. Enfin, lorsqu'un nouveau niveau d'analyse est créé sur une table de dimension, les autres utilisateurs appartenant à la même communauté d'utilisateurs peuvent en tirer profit, dans l'esprit d'un système collaboratif dans lequel chacun apporte sa pierre à l'édifice.

1.3.2 Recommandation interactive de requêtes décisionnelles

L'analyse en ligne commence généralement par la construction de cube données. Cependant, le choix des requêtes correspondant aux cubes OLAP pouvant être intéressants pour l'utilisateur est une tâche fastidieuse qui peut engendrer une perte de temps. Il s'agit de choisir les faits à analyser en termes de mesures et de dimensions en choisissant le bon niveau de granularité pour permettre d'effectuer des analyses OLAP intéressantes pour l'utilisateur. Or, l'utilisateur veut souvent créer de nouveaux cubes OLAP en écrivant de nouvelles requêtes décisionnelles. En effet, un même cube de données n'est pas toujours pertinent pour des tâches d'analyse différentes. Nous pensons que l'utilisateur doit être en mesure de construire les cubes de données qui sont les plus appropriés pour ses tâches d'analyse actuelles. Ceci est important car les besoins des utilisateurs peuvent changer presque quotidiennement et la construction des cubes de données doit bien sûr suivre ces changements.

Ainsi, l'objectif de notre deuxième contribution est d'impliquer davantage l'utilisateur dans le processus d'analyse en ligne en lui recommandant de façon interactive

des requêtes décisionnelles. Les travaux de la littérature s'appuient uniquement sur des requêtes utilisateurs, mais excluent l'utilisateur lui-même [GMN09, JRTZ09b]. Très peu de ces travaux proposent d'accompagner le décideur durant tout le processus d'aide à la décision. Ils sont pour la plupart des systèmes, "intelligents" ou non, cherchant à optimiser des solutions sans être réellement interactifs. Dans ce contexte, nous proposons une approche de recommandation de requêtes décisionnelles pour la construction de cubes OLAP pertinents qui s'appuie sur la charge de requêtes associée à un ou un ensemble d'utilisateurs appartenant à la même communauté d'acteurs d'une organisation. Notre objectif est de réduire le niveau de connaissances techniques dont l'utilisateur a besoin au moment de créer un cube OLAP. Notre intuition est que la pertinence d'une requête décisionnelle est fortement corrélée avec la fréquence d'utilisation par l'utilisateur (ou un ensemble d'utilisateurs) des attributs associés à l'ensemble de ses (leurs) requêtes précédentes. Notre approche de formulation de requêtes décisionnelles est collaborative puisqu'elle permet à l'utilisateur de soumettre à l'entrepôt de données des requêtes pertinentes construites, pas à pas, à partir des attributs les plus fréquemment utilisés par l'ensemble des acteurs de la communauté d'utilisateurs à laquelle il appartient.

L'originalité de notre proposition réside, d'une part, dans son aspect interactif et, d'autre part, dans l'utilisation de la fouille de données, et plus précisément la méthode d'extraction des motifs fréquents *Close* [PBTL99], pour extraire les attributs fréquents utilisés par l'utilisateur à partir de ses requêtes précédentes. Par ailleurs, pour partager l'expérience des autres utilisateurs, notre approche exploite l'historique des requêtes de l'ensemble des utilisateurs du système décisionnel. Pour terminer, nous avons développé un système de recommandation de requêtes décisionnelles interactif et collaboratif, FIMIOQR (Frequent Itemsets Mining for Interactive OLAP Query Recommendation).

1.3.3 Recommandation de chemins de navigation

Lors de la navigation dans un cube de données, l'utilisateur n'a pas une connaissance a priori des parties du cube susceptibles d'être intéressantes pour lui. De ce fait, choisir les prochaines navigations devient une tâche difficile pour l'utilisateur puisque plusieurs chemins de navigation se présentent à lui. Cela pourrait provoquer des temps de latence de l'analyse voire de conduire l'utilisateur dans une zone non pertinente, et de réduire ainsi les avantages de l'utilisation du système OLAP.

Ceci peut s'expliquer par le fait que l'analyse OLAP ne permet pas de prendre en compte le profil de l'utilisateur, en particulier ses usages. Pour pallier ce problème, nous développons une approche centrée utilisateur qui suggère à l'utilisateur des conseils de navigation dans le cube de données. Dans un cube OLAP, les données sont modélisées suivant de multiples dimensions où les sous-cubes sont généralement représentés comme des éléments de treillis de cuboïdes [HRU96]. C'est pourquoi, nous utilisons la structure de treillis de cuboïdes pour décrire les chemins de navigation d'un utilisateur. Les utilisateurs naviguent ainsi d'un cuboïde à l'autre au sein d'un cube OLAP. Le graphe de navigation est donc le treillis de tous les cuboïdes. En effet, la structure de treillis de cuboïdes permet de visualiser tous les chemins de navigation d'un utilisateur et permet surtout de garder le séquencement logique de navigation.

De ce fait, nous assimilons la notion de session d'analyse à un chemin de navigation dans le treillis de cuboïdes correspondant au cube OLAP. Dans une session d'analyse, l'utilisateur passe d'une requête d'analyse à une autre qui lui est directement liée formant ainsi un enchainement de requêtes appelé chemin de navigation. La requête d'analyse suivante ne dépend que de la requête d'analyse en cours. De ce fait, afin de guider l'utilisateur vers un chemin de navigation pertinent pour lui, notre idée est d'utiliser les chaines de Markov qui ont la propriété suivante : "Le futur ne dépend que de l'état présent". Autrement dit, il est possible de prédire l'état suivant avec la seule connaissance de l'état présent. Ainsi, à partir d'un point de navigation (nœud courant dans le treillis de cuboïdes), et en utilisant la propriété des chaînes de Markov, il devient possible de prédire le point de navigation suivant à l'utilisateur.

Ce travail s'inscrit dans une approche de recommandation collaborative car les probabilités de passage d'une requête à l'autre dans le treillis de cuboïdes est calculé en tenant compte des toutes les requêtes d'analyse de l'ensemble des utilisateurs appartenant à la même communauté. Nous avons conçu pour cela un système de recommandation de chemins de navigation nommé NAPARE (NAvigation PAth REcommendation) qui s'appuie sur le modèle de Markov construit à partir de l'historique des sessions des analyses de l'ensemble des utilisateurs exploitant le même cube OLAP.

1.4 Organisation du mémoire

Pour présenter nos différentes contributions, nous avons retenu pour ce mémoire de thèse une organisation en sept chapitres. Après un premier chapitre qui est consacré à l'introduction générale de nos travaux, le chapitre 2 est dédié à l'état de l'art relatif aux travaux de recherche concernant la personnalisation dans les bases de données et dans les entrepôts de données. Nous présentons une classification et une analyse critique de ces travaux selon un ensemble de critères que nous avons définis. Le chapitre 3 présente notre approche de personnalisation du contenu de l'entrepôt de données. Nous évoquons les objectifs et les motivations qui nous ont poussés à adapter le contenu de l'entrepôt de données en fonction du profil utilisateur. Ce chapitre inclut aussi une étude de cas, ainsi que des résultats expérimentaux évaluant la performance de nos méthodes. Nous développons dans le chapitre 4 notre approche de recommandation interactive de requêtes décisionnelles. Nous montrons l'intérêt de notre approche et motivons le choix de l'utilisation de la fouille de données comme technique d'extraction du profil utilisateur. Nous détaillons les principes de notre démarche, notamment la construction du contexte d'extraction et l'algorithme de recommandation. Nous présentons également deux mesures de qualité pour l'évaluation de nos méthodes de recommandation. Dans le chapitre 5, nous abordons notre approche de recommandation des chemins de navigation en exploitant l'historique des requêtes (journal de requêtes). Nous détaillons les principaux travaux ayant intégré l'aspect navigationnel dans les entrepôts de données. Nous décrivons notre cadre général concernant la navigation guidée dans les cubes de données. Nous présentons également l'algorithme de recommandation basé sur le modèle de Markov. Des résultats expérimentaux sont aussi fournis afin d'apprécier les performances de notre

algorithme. Pour valider nos travaux, le chapitre 6 présente l'implémentation de la plateforme logicielle *PersoReco* (**Perso**nnalisation et **Reco**mmandation) ainsi que le cas d'application à l'entrepôt de données *Foodmart*[1]. Nous décrivons l'architecture générale de notre plateforme logicielle et détaillons ensuite ses modules prévus pour nos différentes approches de personnalisation dans les entrepôts de données. Nous exposons les différentes étapes de l'implication de l'utilisateur et nous discutons les résultats obtenus. Enfin, le chapitre 7 conclut ce mémoire en présentant un bilan général de l'ensemble de nos contributions en évoquant de nouvelles perspectives de recherche.

1. http ://www.reportportal.com/forum

Chapitre 2

État de l'art

Nous présentons dans ce chapitre un état de l'art sur les travaux relatifs à la prise en compte de l'utilisateur dans les systèmes d'information, et plus particulièrement dans les bases de données et les entrepôts de données. Ces dernières années, la prise en compte de l'utilisateur dans les bases de données ou les entrepôts de données a fait l'objet de nombreuses recherches [KI04b, SP08, BFB08, JRTZ09a, GMNS09, GR09], sans pour autant avoir reçu à l'heure actuelle de solution générale. Ainsi, nous présentons une revue de ces travaux de recherche en faisant un tour d'horizon, non exhaustif, des systèmes de personnalisation et de recommandation liés aux domaines des bases de données et des entrepôts de données, en évoquant leur origine et leurs applications et en décrivant les données qu'ils exploitent. De plus, nous présentons les principales techniques de personnalisation et de recommandation en soulignant leurs apports et leurs limites et discutons les principaux verrous scientifiques auxquels nous nous intéressons particulièrement dans le cadre de cette thèse. Ainsi, l'objectif poursuivi par cet état de l'art est donc d'établir une classification des divers concepts constitutifs de la personnalisation et de la recommandation, disséminés dans la littérature, afin de fournir une base suffisamment solide pour élaborer des analyses OLAP centrées-utilisateur.

2.1 Introduction

Les systèmes d'accès à l'information ont pour objectif de fournir des résultats répondant aux besoins et attentes des utilisateurs et ne nécessitant aucun effort et traitement supplémentaires pour être examinés et exploités. En ce sens, deux problèmes distincts sont fréquemment soulevés. Le premier est comment gérer les requêtes retournant un résultat vide. Si aucun résultat ne répond pleinement à la requête soumise par l'utilisateur, l'enjeu est alors de fournir des réponses alternatives (recommandations). Le deuxième problème concerne les requêtes retournant une surabondance de résultats. Ces requêtes retournent un nombre de réponses trop important pour être exploitées efficacement et facilement par l'utilisateur. Cette situation problématique est d'autant plus actuelle que les bases de données sont de plus en plus volumineuses. Parmi l'ensemble volumineux des résultats retournés, l'utilisateur n'est généralement intéressé que par le sous ensemble des résultats qu'il juge les

plus pertinents. Pour extraire ce sous ensemble, l'utilisateur doit parcourir la totalité des résultats retournés, ce qui constitue pour lui une tâche fastidieuse et malaisée.

La solution consiste donc à faire coopérer le système avec l'utilisateur. Ce dernier peut exprimer ses préférences relatives aux résultats qu'il souhaite obtenir. Le système, en se basant sur les préférences utilisateur, va réduire l'ensemble des résultats obtenus et ne retourne que ceux qui sont pertinents comme le confirme Rizzi *"Expressing preferences when querying databases is a natural way to avoid empty results and information flooding, and in general to rank results so that the user may first see the data that better match his tastes"* [Riz07]. Ainsi, nous pouvons dire que la prise en compte de l'utilisateur dans un système d'information a pour objectif principal l'amélioration de l'extraction des informations en fonction de la perception et des intérêts de cet utilisateur. Cette extraction implique l'utilisation des techniques de personnalisation et de recommandation en se basant sur les préférences des utilisateurs, de ses usages précédents, de ses contraintes, etc.

L'objectif de ce chapitre est de présenter un état de l'art sur la prise en compte de l'utilisateur dans les bases de données et plus spécifiquement dans les entrepôts de données. A cet effet, la section 2.2 est consacrée à la définition des notions de personnalisation et de recommandation. Ensuite, nous présentons les différentes étapes du processus de prise en compte de l'utilisateur car cela influe sur la présentation des différentes approches. Puis, les approches existantes qui impliquent plus l'utilisateur dans les bases de données sont présentées et comparées dans la section 2.4.2. Enfin, les approches existantes de prise en compte de l'utilisateur dans les entrepôts de données sont présentées et comparées dans la section 2.4.2.

2.2 Prise en compte de l'utilisateur dans les systèmes d'information

Dans les systèmes d'information, la prise en compte de l'utilisateur permet d'afficher un contenu informationnel pertinent vis-à-vis des centres d'intérêt des utilisateurs. Ainsi, l'avantage pour l'utilisateur est qu'il ne sera pas inondé d'informations sans relation avec ses attentes mais au contraire il aura des informations correspondant au mieux à ses centres d'intérêt. Il existe deux types d'approches pour la prise en compte de l'utilisateur, à savoir la personnalisation et la recommandation.

2.2.1 Personnalisation

Le terme "personnalisation" a connu un grand succès ces dernières années. Il a attiré l'attention des concepteurs des sites Internet, de logiciels ou encore de produits de consommation. Il est utilisé pour décrire la pratique de l'affichage du contenu destiné à un utilisateur ou un groupe d'utilisateurs. En effet, la personnalisation a été utilisée afin de faire face au problème de surcharge et de profusion d'informations disponibles notamment à travers le Web [FK00, MGSG07], le e-commerce [AGPS02, KA08, BNG+], la RI (recherche d'information) [BRS00, FdS01, VCF+07], les bases de données [Cho03, KI04b, GRB11] et même les entrepôts de données [BGM+05, BFB08].

La personnalisation consiste à utiliser les données comportementales recueillies sur l'utilisateur pour cibler ses intérêts. Il s'agit donc de fournir à l'utilisateur un contenu créé en amont en fonction de son profil.

Ainsi, la personnalisation se retrouve dans tant de travaux de recherches traitant pourtant des sujets très différents. Suite à cet engouement, nous constatons une difficulté croissante à en saisir la signification exacte ce qui nous incite à clarifier ce vaste sujet. Ainsi, nous nous intéressons à définir ce que c'est la personnalisation après une revue des définitions qui existent dans la littérature. Par ailleurs, l'étude de la littérature a montré une certaine ambiguïté quant à la définition de la personnalisation et de la recommandation. Alors, il est temps de faire la distinction entre la recommandation et la personnalisation pour mieux comprendre les problèmes posés et les solutions proposées.

Le terme personnalisation est utilisé pour expliquer comment recevoir à partir d'une grande quantité d'informations seulement la partie qui intéresse un individu ou un groupe d'individus. Pour connaître ce qui intéresse l'utilisateur, il faut connaître son profil (ses intérêts, ses préférences ou même ses contraintes, ses comportements, etc.). Le terme personnalisation est alors utilisé pour la prise en compte du profil utilisateur.

De ce fait, aujourd'hui, on utilise les systèmes personnalisés qui tiennent compte des besoins, des intérêts et des préférences individuels des utilisateurs. Les utilisateurs s'attendent qu'on exploite l'information qu'on a sur eux et de continuer à apprendre à partir de leur relations avec le système (comportement). Ils ne veulent pas perdre leur temps avec des informations inappropriées ou non pertinentes.

La personnalisation à l'accès à l'information est définie comme un processus qui change la fonctionnalité, l'interface, la teneur en information, ou l'aspect d'un système pour augmenter sa pertinence personnelle en fonction des caractéristiques sociodémographiques déclarées de l'utilisateur (sexe, âge, lieu de résidence, etc.) et/ou de son comportement observé. D'après Turpeinen [TS04], la personnalisation est définie comme suit : *"Personalization is a process that changes the functionality, interface, information content, or appearance of a system to increase its personal relevance to an individual. Personalization systems accommodate individual's needs and interests explicitly through changes and selections initiated by the user, and implicitly through automatic adaptation techniques"*.

De ce fait, la personnalisation vise à répondre de façon adaptée aux besoins et aux caractéristiques de chaque utilisateur. Il s'agit de gérer la connaissance relative à l'utilisateur et exploiter cette connaissance pour décider de ce qui doit être présenté à l'utilisateur.

Par ailleurs, Ioannidis et Koutrika [IK05] définissent aussi la personnalisation comme *"...providing an overall customized, individualized user experience by taking into account the needs, preferences and characteristics of a user or group of users"*. Ainsi, la personnalisation de requêtes est considérée comme un ajout de contraintes, issues du profil de l'utilisateur, à une requête donnée [KI04b, BGM+05].

A la lumière de ces définitions, nous pouvons dire que la personnalisation a pour objectif de faciliter l'expression des besoins des utilisateurs et de rendre l'information sélectionnée intelligible à l'utilisateur et exploitable. Ainsi, la personnalisation permet de placer l'utilisateur au cœur du système. Ceci est fait par l'exploitation

d'un ensemble de connaissances, de préférences individuelles, des ordonnancements de critères ou des règles sémantiques spécifiques à chaque utilisateur ou communauté d'utilisateurs. Ces modes de spécification servent à décrire le profil utilisateur permettant de fournir des réponses pertinentes. Par conséquent, la fonction de personnalisation d'un système peut être résumée en deux phases : (i) définition du profil et (ii) exploitation du profil afin de fournir des réponses pertinentes à l'utilisateur. Le système doit définir le profil de l'utilisateur d'une manière implicite ou explicite puis l'exploiter que ce soit par adaptation du système ou par recommandation de nouvelles informations à l'utilisateur.

Par exemple, nous considérons deux décideurs couvrant deux rôles différents mais partageant le même entrepôt de données. Le directeur des ventes est essentiellement intéressé par les ventes mensuelles, mais il peut également souhaiter voir des données plus détaillées si la vente hebdomadaire dépasse un certain montant. Par contre, un agent de ventes travaillant dans une région donnée préfère voir les données concernant les commissions les plus élevées dans cette région. Ayant des intérêts différents et plusieurs objectifs d'analyse des données, les deux utilisateurs ont besoin d'un grand effort et de temps perdu dans la formulation des requêtes pour atteindre leurs objectifs. Ainsi, les deux utilisateurs de l'entrepôt de données peuvent obtenir des résultats différents pour la même requête dans le cas où leurs préférences peuvent être exploitées durant la phase d'analyse via un processus de personnalisation de l'entrepôt de données.

2.2.2 Recommandation

Tout d'abord, nous précisons la différence entre personnalisation et recommandation. En effet, la personnalisation et la recommandation constituent deux notions très liées. Toutefois, la personnalisation évoque les préférences personnelles tandis que la recommandation évoque des suggestions sans recours obligatoirement aux préférences utilisateur. Par ailleurs, les recommandations peuvent être proposées en fonction du comportement général de l'utilisateur. Selon notre point de vue, nous considérons la recommandation comme une forme particulière de la personnalisation qui permet d'anticiper le choix de l'utilisateur. Ainsi, nous pouvons personnaliser via la recommandation.

L'essor du Web et sa popularité ont engendré la mise en place des systèmes de recommandation dans de nombreux domaines : la recherche d'information [BYRN99, CCDM13, SBdVK13], le e-commerce [SKR99, PZZ09], le web usage mining [CKK02, BHMD05, SS09], les bases de données [SDP09, YPS09], les entrepôts de données [BF09, GMNS09, JRTZ09a] et bien d'autres. En effet, un système de recommandation a pour objectif de fournir à un utilisateur un contenu pertinent en fonction de ses préférences Ce dernier peut ainsi réduire son temps de recherche et reçoit également des suggestions de la part du système auxquelles il n'aurait pas prêtées attention.

Afin de fournir une recommandation adaptée, les systèmes de recommandation analysent les profils des utilisateurs et des objets. De façon générale, ils se basent sur les notes déjà fournies par les utilisateurs sur les objets. Ces notes peuvent être fournies explicitement par les utilisateurs ou être calculées par le système en fonction de certaines traces d'utilisation par exemple (articles consultés, nombre de clics,

durée, etc.). Ainsi, les systèmes de recommandation sont des systèmes d'information de filtrage visant à représenter les éléments d'information qui peuvent intéresser l'utilisateur. Trois approches de recommandation sont présentes dans la littérature : l'approche de recommandation basée sur le contenu (content based recommendation) qui repose sur le contenu seulement, l'approche de filtrage collaboratif (collaborative filtering recommendation) tente de trouver des liens ente les utilisateurs via leurs intérêts respectifs et l'approche de recommandation hybride qui combine les deux.

2.2.2.1 Recommandation cognitive

Les systèmes de recommandation basés sur le contenu s'appuient sur des évaluations effectuées par un utilisateur sur un ensemble d'objets (documents, livres, films, etc.). L'objectif est alors de comprendre les motivations l'ayant conduit à juger comme pertinent ou non un objet donné. Le système peut alors proposer à l'utilisateur un choix parmi de nouveaux objets jugés proches des objets qu'il a précédemment appréciés.

Les systèmes de recommandation basés sur le contenu comparent les caractéristiques des objets afin de recommander des objets similaires à ceux déjà appréciés par l'utilisateur.

2.2.2.2 Recommandation collaborative

La notion de filtrage collaboratif est à la base de la recommandation, les méthodes de filtrage par le contenu étant plutôt liées aux systèmes de recherche d'informations personnalisés. Elle se fonde non plus sur le même utilisateur mais cherche à rapprocher l'utilisateur courant avec un ensemble d'utilisateurs existants. Le filtrage collaboratif se base sur l'idée que les personnes à la recherche d'information devraient se servir de ce que d'autres ont déjà trouvé et évalué.

Généralement, dans la recommandation collaborative, le problème de base est le suivant : on dispose d'un ensemble de m objets (documents, films, etc.), de n utilisateurs et d'une matrice d'utilité $R = (r_{ij}, i = 1, ...n, j = 1, ..m)$ (utility matrix ou bien rating matrix) telle que :

- $r_{ij} \in R$ signifie que l'utilisateur i a attribué la note r_{ij} à l'objet j ;
- $r_{ij} = *$ signifie que la note de l'utilisateur i à l'objet j n'est pas connue.

Les approches de filtrage collaboratif comparent les utilisateurs en fonction des notes déjà obtenues et agrègent les notes des utilisateurs similaires afin de proposer leurs objets préférés.

2.2.2.3 Recommandation hybride

Les systèmes de filtrages hybrides combinent les deux types de filtrage : le filtrage basé sur le contenu et le filtrage collaboratif. Ainsi, la recommandation personnalisée utilise les données comportementales de l'utilisateur pour personnaliser en temps réel le contenu fourni. Les systèmes de recommandation hybride fournissent des suggestions en adéquation avec le profil utilisateur. Par exemple, cette technique est utilisée en e-commerce pour proposer une offre de produits unique et spécifique à

l'internaute. Cette technique consiste à filtrer les recommandations par les préférences personnelles pour avoir des recommandations personnalisées.

Les approches hybrides utilisent des algorithmes alliant des comparaisons entre profils d'objets et profils d'utilisateurs.

2.3 Processus de prise en compte de l'utilisateur dans les systèmes d'information

La prise en compte de l'utilisateur dans un système d'information doit suivre un processus bien défini qui peut différer d'un domaine à l'autre mais qui se présente généralement comme illustré dans la Figure 2.1. En effet, la prise en compte de l'utilisateur dans un système d'information comporte deux grandes phases à savoir (1) la définition du profil utilisateur et (2) l'exploitation de ce dernier. La première phase se réalise par la collecte d'information concernant l'utilisateur. Cette collecte peut être implicite ou explicite. La deuxième phase à savoir l'exploitation du profil utilisateur déjà défini peut se dérouler par deux manières différentes : la personnalisation ou la recommandation. Ainsi, nous détaillons dans la suite toutes les étapes du processus de prise en compte de l'utilisateur dans un système d'information.

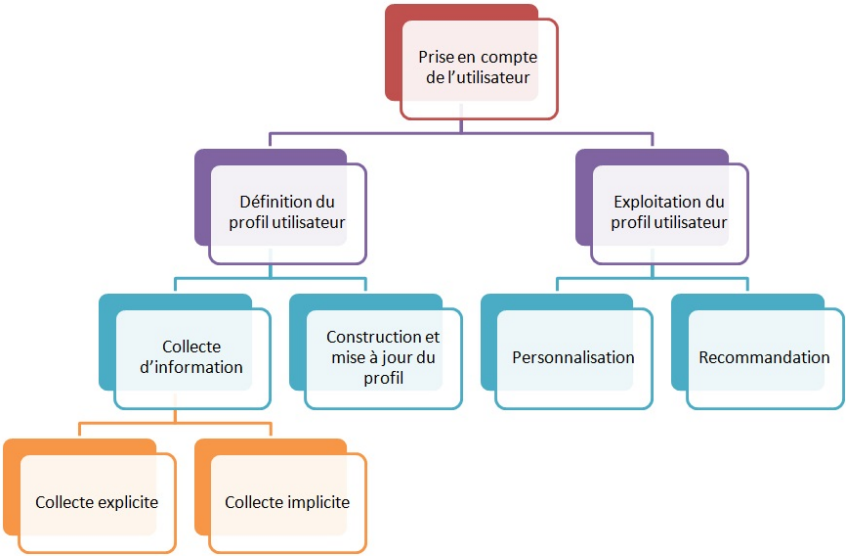


Figure 2.1 – Processus de personnalisation

2.3.1 Collecte d'information

Pour tous les systèmes de personnalisation développés jusqu'à nos jours, la collecte de données relatives aux utilisateurs représente une phase clé dans le processus de personnalisation. Il s'agit de déterminer comment sont collectées les informations liées aux utilisateurs et à leurs besoins. Cette collecte peut se faire de manière explicite ou implicite.

Collecte explicite. La collecte explicite d'information correspond à toute donnée qui a été saisie ou fournie directement par l'utilisateur. On l'appelle aussi déclaration puisque c'est à l'utilisateur de déclarer ses informations. C'est la méthode la plus directe pour acquérir des informations sur l'utilisateur. Généralement, le mode d'acquisition explicite des données est le plus facile à mettre en œuvre. En effet, au niveau le plus basique, l'utilisateur est invité à saisir manuellement des informations utilisées dans la construction de son profil. Vraisemblablement, la demande explicite peut être le meilleur moyen pour recueillir des informations de bonne qualité, qui peuvent refléter les besoins subtils et les préférences des utilisateurs. Cependant, la collecte explicite est une tâche fastidieuse pour l'utilisateur qui a tendance à être réticent quant à l'intervention directe pour exprimer ses intérêts. L'effort supplémentaire imposé à spécifier explicitement ses besoins est indésirable pour les utilisateurs. Ils peuvent de plus avoir l'impression que le processus prend trop de temps pour une amélioration peu perceptible du service offert par le système. Certains utilisateurs développent une crainte d'être surveillés et évitent les systèmes les incitant à fournir des données, ou laissant entendre que leur comportement est traqué pour personnaliser le contenu. La crainte d'une violation de leur vie privée les empêche de fournir des informations. D'autre part, si l'utilisateur fournit ses informations, la valeur de ce type d'information n'est pas toujours fiable car rien ne garantit la véracité de ses informations.

Collecte implicite. Par opposition à la collecte explicite, la collecte de l'information peut se faire implicitement sans la sollicitation directe de l'utilisateur. Ainsi, les informations sont récoltées sans que l'utilisateur n'ait besoin de préciser ces informations (comme par exemple les traces d'usage dans les fichiers logs).

Dans ce type de collecte d'information, l'observation des interactions de l'utilisateur avec le système est indirecte, elle repose sur des données ou des appréciations "implicites" déduites à partir des actions réalisées par cet utilisateur. Ces actions sont souvent appelées "les traces d'usage". Ces traces représentent une suite d'actions effectuées par un utilisateur, elles sont souvent déduites à partir des fichiers logs.

Ainsi, il est possible d'enrichir les informations déclaratives des utilisateurs grâce à l'enregistrement de données par des moyens techniques. Le procédé de collecte d'informations le plus important concerne en effet les fichiers log qui constituent une source importante des données de l'utilisateur. En effet, dans le contexte des bases de données, l'exploitation des logs de requêtes par exemple peut aider à personnaliser les requêtes futures des utilisateurs ou à proposer des recommandations [KBG+09, CEP09, GMN09, SDP09].

Le mode d'acquisition implicite repose sur des techniques d'extraction des informations basées sur des mesures de pertinence implicite (fréquence, temps d'exploration, etc) appliquées sur l'historique d'interactions de l'utilisateur.

Par ailleurs, la démarche de collecte des données explicites ou implicites (les traces d'usage) dans le cadre d'un système de personnalisation doit veiller à la mise à jour de des informations collectées. Ainsi, la démarche de collecte relève plus d'efficacité si le profil utilisateur est enrichi au fil de ses activités dans le système. Pour remplir cet objectif, il est nécessaire de définir des solutions permettant l'évolution des modèles utilisateurs et leur extensibilité.

2.3.2 Construction du profil utilisateur

La personnalisation et/ou la recommandation nécessitent la connaissance de l'utilisateur qui est exprimée sous forme d'un modèle le décrivant. Dans la littérature, ce modèle est une collection de diverses informations (données personnelles, données démographiques, préférences, centres d'intérêts, compétences, usages, etc.) collectées de façon explicite et/ou implicite.

Après une revue de la littérature, nous pouvons dire que les informations collectées peuvent être diverses. Il y a d'abord ce que l'utilisateur connaît déjà (Knowledge background). Cela permet d'effectuer une évaluation préalable des connaissances de l'utilisateur (adaptation statique). Puis, vient l'évaluation dynamique des connaissances pour améliorer le processus de personnalisation. Ensuite, en se basant sur l'identité de l'utilisateur (donnés personnelles), ses connaissances, ses intérêts spécifiques, ..., ses buts peuvent être identifiés. En outre, le tracking du comportement de l'utilisateur peut avoir lieu. En effet, les traces d'usages (souvent dans le fichier log) représente le comportement de l'utilisateur. Enfin, la notion de contexte peut comporter des informations aussi diverses que la date, l'emplacement de l'utilisateur, les caractéristiques matérielles et logicielles du dispositif d'accès, etc. En général, une donnée est pertinente si et seulement si elle satisfait les contraintes utilisateur. Parfois, pour être pertinentes, les données doivent respecter le contexte de l'utilisateur. Cependant, le contexte utilisateur représente un grand problème : Comment le représenter ? Il est représentable pour certains "Context can be represented and processed" [ADB+99] et non représentable pour d'autres [Dou04].

La modélisation dans un tel processus est une étape primordiale. Cependant, cette étape peut devenir difficile et complexe par le manque de données. En effet, les données relatives aux utilisateurs ne sont pas toujours disponibles dans le système.

Dans la littérature, le modèle utilisateur est centré sur plusieurs notions dont les définitions sont floues : profil, contexte, préférences, etc. Ainsi, il manque un consensus entre les différentes approches puisque la représentation de l'utilisateur varie d'une approche à l'autre. Dans le domaine de la RI (Recherche d'Information) par exemple, une solution intéressante a été proposée par Bouzeghoub et Kostadinov qui présentent un modèle générique multidimensionnel qui couvre la majorité des informations caractérisant l'utilisateur [BK05].

Dans le contexte des bases de données, il n'existe pas de notion de profil utilisateur, mais une extension des langages de requêtes avec des opérateurs ou des clauses supportant des préférences. Plus particulièrement, il existe des langages et opérateurs de préférence qui ont été proposés pour satisfaire les préférences utilisateur avec une extension de SQL, appelée PRFERENCE SQL [KK02]. Les opérateurs WINNOW [Cho02] et SKYLINE [BKS01] ou même des clauses telles que PREFER ou FROM WHICH PREFER [LL87] existent.

En s'appuyant sur les différentes propositions relatives au modèle utilisateur qui existent dans la littérature, notre premier constat est de dire que , souvent, le "modèle utilisateur" est confondu au "profil utilisateur". Cependant, le profil utilisateur n'est qu'une instance du modèle utilisateur. En effet, le profil utilisateur présente une implémentation du modèle utilisateur qui peut varier selon le domaine d'application. Par exemple, il est représenté sous forme de mots clés en RI (Recherche d'Infor-

mation) [FdS01] tandis qu'il est représenté sous forme de conditions de sélection ou de jointures de requêtes en bases de données [KI04b]. Notre deuxième constat est que quelque soit l'approche proposée, le concept de préférence revient constamment. Une préférence est une expression permettant de hiérarchiser l'importance des informations dans un profil. Ainsi, le profil utilisateur est réduit à ses préférences. Par conséquent, un profil est un recueil de préférences utilisateur sur les données. Les données du profil utilisateur peuvent être saisis manuellement par l'utilisateur, ou collectées implicitement à partir de ses interactions précédentes avec le système.

Dans ce contexte, les préférences utilisateur constituent la clé de la prise en compte de l'utilisateur dans les systèmes décisionnels. En effet, les décisions ne sont pas indépendantes de "intérêts" ou "valeurs" particuliers. De ce fait, dès lors que l'on se préoccupe de "décision", s'intéresser à la modélisation des préférences semble inévitable puisqu'il faut penser à modéliser comment comparer en termes de préférence les objets de la décision.

On appelle "préférence utilisateur", un ensemble de descriptions englobant : ce qu'un utilisateur envisage d'accomplir dans le système, la manière de le faire, le type et l'ordre des résultats qu'il souhaite et la manière dont il aimerait que l'information soit affichée. Les préférences utilisateur correspondent à un ensemble de critères permettant pour un utilisateur spécifique de mesurer la pertinence d'une information et d'évaluer si une information est plus pertinente qu'une autre information.

La notion de préférence de requête (preference query) a été introduite dans le domaine des bases de données la première fois par Lacroix [LL87]. Il extencie le domaine de calcul relationnel Domain relationnal Calculus pour exprimer les préférences pour les tuples satisfaisant certaines conditions logiques.

A partir de son introduction, plusieurs investigations extensives ont été menées et deux grandes lignes des approches ont été formées dans la littérature pour traiter les préférences utilisateur appelées quantitative et qualitative [Cho03].

L'approche qualitative est destinée à spécifier directement les préférences entre les éléments dans la réponse à la requête, typiquement en utilisant les relations de préférences binaires. Un exemple de relation de préférence est : " préférer un tuple de produit à un autre s'ils ont la même référence mais le prix du premier est moins cher que celui du deuxième". Ces types de relations de préférences peuvent être intégrés dans les langages relationnels de requête par des opérateurs relationnels ou des constructeurs spéciaux pour la préférence qui sélectionnent de leur entrée l'ensemble des tuples les plus préférés (WINNOW [Cho03], PREFERENCE SQL [KK02] et SKYLINE [BKS01].

L'approche quantitative exprime les préférences en utilisant les fonctions de scores, qui associent un score numérique à chaque élément. Par exemple, le tuple t_1 est préféré au tuple t_2 si et seulement si le score de t_1 est plus grand que t_2.

Par conséquent, la formulation qualitative est intuitive pour l'utilisateur. Cependant, les approches quantitatives permettent de formuler des préférences d'une manière absolue sur les éléments désirés. En terme d'expressivité, les approches qualitatives sont plus générales puisqu'on ne peut pas traduire toutes les relations d'ordre par des fonctions de score. Cependant, ces approches ne permettent pas de traduire la différence de l'intensité des préférences.

Une plateforme pour exprimer et combiner ces types de fonctions de préférence

est fournie en [GR09] où un modèle de préférence plus riche qui peut associer les degrés d'intérêts (comme score) avec les préférences dans un schéma de base de données est présenté.

Souvent, les centres d'intérêt des utilisateurs peuvent varier au cours du temps ce qui nécessite un effort de suivi. De ce fait, un processus complémentaire à la construction du profil utilisateur s'impose. Il s'agit de la gestion de l'évolution de ce profil qui consiste à le mettre à jour en fonction des variations détectées. Cette mise à jour consiste principalement à deux phases à savoir (1) la capture des changements des centres d'intérêt de l'utilisateur et (2) la propagation de ces changements au niveau de la représentation du profil. L'évolution du profil utilisateur se fait souvent selon un processus incrémental basé sur l'addition de nouvelles informations dans la représentation du profil [DTBC08].

2.3.3 Exploitation du profil utilisateur

Afin de permettre à un processus décisionnel d'être centré utilisateur, la prise en compte des préférences et des caractéristiques des utilisateurs à travers leur profil constitue la clé de personnalisation. Il s'agit alors d'adapter le système (contenu, visualisation, etc) en fonction du profil utilisateur et faire de la recommandation d'informations pertinentes.

L'adaptation peut être assurée par deux techniques différentes à savoir (1) filtrage des données ou des résultats et (2) tri des résultats. En effet, la requête peut être affinée avant son exécution afin d'avoir directement un résultat pertinent [KI04b, KI05], ou exécuter la requête puis réduire le résultat en présentant seulement l'information pertinente [BRS00]. En outre, les résultats peuvent être triés afin de présenter les informations les plus pertinentes en premier lieu [SLC+08].

La deuxième forme de prise en compte de l'utilisateur dans les systèmes d'information est la recommandation qui consiste à proposer à l'utilisateur des données qui peuvent l'intéresser et surtout utiliser ses préférences pour avoir des recommandations personnalisées.

Dans ce qui suit, nous allons nous focaliser sur les travaux existants dans la littérature relatifs à la personnalisation dans les entrepôts de données. Ces travaux sont largement inspirés des travaux réalisés dans le domaine des bases de données.

2.4 Les travaux existants de prise en compte de l'utilisateur dans les bases de données et entrepôts de données

Il existe deux catégories de travaux. La première catégorie concerne la personnalisation où les travaux visent à adapter le système en fonction des préférences utilisateur explicitement ou implicitement collectées. La deuxième catégorie est orientée plutôt vers la recommandation où les travaux consistent à recommander souvent des requêtes en se basant sur les préférences ou l'historique des requêtes de l'utilisateur ou d'un groupe d'utilisateurs.

2.4.1 Critères d'étude des travaux de prise en compte de l'utilisateur

Pour bien comprendre les enjeux de la prise en compte de l'utilisateur dans les bases et les entrepôts de données, nous présentons tout d'abord des critères que nous avons jugés pertinents pour étudier les différentes facettes de de la personnalisation et de la recommandation. Nous présentons une synthèse de ces travaux que ce soit dans les base de données (Tableau 2.1) ou dans les entrepôts de données (Tableau 2.2).

Nous avons défini les critères suivants :

- Rôle : ce critère représente l'objectif des travaux. En effet, les travaux distinguent bien la personnalisation de la recommandation.

- Type : ce critère représente le type de système de personnalisation/recommandation utilisé. Nous distinguons deux types de personnalisation/recommandation : la personnalisation/recommandation cognitive et la personnalisation/recommandation sociale ou collaborative. Ces deux types de personnalisation/recommandation peuvent être combinés. La personnalisation/recommandation cognitive permet de prendre en compte l'utilisateur de façon individuelle, en fonction de ses besoins. Ce type de personnalisation/recommandation est basé sur le profil de l'utilisateur ou sur l'analyse des données contextuelles manipulées par l'utilisateur. Cependant, la personnalisation/recommandation sociale permet la prise en compte du contexte des autres utilisateurs qui auraient des préoccupations similaires.

- Profil : ce critère représente la source de personnalisation/recommandation qui peut être sous forme de préférences utilisateur, d'usage (historique des requêtes : fichier log) ou des sources externes.

- Moment de la personnalisation/recommandation : le moment où la personnalisation/recommandation est réalisée par rapport à l'interrogation : avant, pendant ou après la soumission de la requête.

- Collecte de l'information : la collecte d'information peut être implicite ou explicite (Section 5.3.2.

- Formulation des préférences : la formulation des préférences peut être qualitative ou quantitative (Section 5.3.2).

- Objet d'exploitation : L'objet à exploiter peut être le schéma de la base de données/entrepôt de données, la requête/fragment de requêtes ou la visualisation (interface).

- Technique utilisée : Même si l'objectif est d'impliquer plus l'utilisateur, les méthodes existantes n'utilisent pas les mêmes techniques. Nous identifions donc la technique utilisée par chaque méthode proposée.

Il va de soi que dans la réalité, tous les concepts présentés en réponse à ces critères sont intimement liés et le plus souvent combinés. Cette dernière caractéristique constitue d'ailleurs l'une des sources du grand potentiel de la personnalisation et de la recommandation : ses multiples facettes peuvent être largement combinées, offrant ainsi aux concepteurs un grand nombre de possibilités de personnalisation/recommandation encore inexploitées à l'heure actuelle.

2.4.2 L'utilisateur dans les bases de données

Dans le domaine des bases de données, la prise en compte de l'utilisateur est fondée généralement sur son profil. Néanmoins, le contenu et l'exploitation de ce profil varie selon les approches.

2.4.2.1 Personnalisation dans les bases de données

L'axe le plus émergent est la personnalisation de requêtes. L'idée principale derrière ce processus de personnalisation consiste à considérer les préférences utilisateur lors de la réponse à la requête [KI05]. En effet, la personnalisation de requêtes dans les bases de données peut être réalisée par un enrichissement de la requête ou par des opérateurs de préférences.

2.4.2.1.1 Personnalisation par enrichissement de requêtes

L'enrichissement de la requête consiste à intégrer des éléments du profil utilisateur dans la requête de l'utilisateur. Cette technique est bien utilisée dans le domaine de la recherche de l'information (RI), elle est relativement récente dans le domaine des bases de données et des entrepôts de données. Dans les travaux de Koutrika et al., le profil utilisateur est composé par un ensemble de préférences obtenues explicitement par l'utilisateur [KI04a]. Chaque préférence est représentée par un prédicat et un score associé à ce prédicat. Nous distinguons deux types de préférences : (i) préférence de sélection et (ii) préférence de jointure. La préférence de sélection est une préférence d'un utilisateur pour une condition de sélection s qui relie l'attribut à une valeur et exprimée par un degré d'intérêt dénoté par $doi(s)$ et défini par $doi(s) =< d >$ et $d \in [0, 1]$. Le degré 0 indique que l'utilisateur ne s'intéresse pas à cette condition de sélection s. Par contre, le degré 1 indique l'intérêt élevé pour cette condition s. La préférence de jointure consiste en une préférence d'un utilisateur pour une condition de jointure q entre les attributs et exprimée par un degré d'intérêt dénoté par $doi(q)$ et défini par $doi(q) =< d >$ et $d \in [0, 1]$. Le degré 0 indique que l'utilisateur ne s'intéresse pas à cette condition de jointure. Par contre, le degré 1 indique l'intérêt élevé pour cette condition q. Par conséquent, le profil utilisateur est défini par comme une liste de prédicats disjoints, incluant les sélections et les jointures. Les prédicats sont associés de poids dans l'intervalle $[0, 1]$ et représentant leur importance relative en respectant les préférences de l'utilisateur [KI04b, KI05]. Avec ce profil utilisateur, le processus de l'enrichissement de la requête consiste à reformuler la requête initiale en ajoutant des prédicats du profil. Il consiste en 2 phases : (i) sélection de prédicats et (ii) intégration des prédicats dans la requête.

2.4.2.1.2 Personnalisation par une algèbre spécifique

Dans la littérature, il existe des langages et opérateurs de préférence qui ont été proposés pour satisfaire plus l'utilisateur.

PREFERENCE SQL. Une extension intéressante de SQL, appelée PRFERENCE SQL [KK02] permet d'exprimer un nombre important de préférences. PREFERENCE SQL propose des opérateurs approximatifs tels que ARROUND et BETWEEN, mais leur définition est fixée. Ainsi, une requête possède le même résultat indépendamment de l'utilisateur qu'il l'a posée. En effet, la préférence "Autour de" (AROUND) exprime l'intérêt de l'utilisateur vers les éléments dont la valeur d'un attribut est la plus proche possible de la valeur donnée en paramètre. De plus, la préférence "Entre" (BETWEEN) permet à l'utilisateur d'exprimer ses préférences pour les éléments dont la valeur pour un attribut donné est comprise dans un intervalle, les valeurs extrêmes (low et up) sont les meilleurs. Une autre approche pour l'expression de préférences dans les critères de sélection est utilisée dans PREFERENCE SQL [KK02, Kie05]. Elle permet la définition des ensembles de valeurs positifs (POS) ou négatifs (NEG) pour un attribut donné et si ces ensembles sont insuffisants, une préférence explicite (EXPLICIT) dans un couple de valeur peut être exprimée. En utilisant le langage PREFERENCE SQL, nous pouvons aussi définir des hiérarchies de préférences qui spécifient un ordre partiel entre les critères de sélection en utilisant 2 clauses PREFERING et CASCADING. La première clause (PREFERING) fixe le choix initial de critères de sélection, c'est-à-dire les critères qui vont être considérés immédiatement après les sélections de la requête. La deuxième clause (CASCADING) introduit les critères qui sont moins important que les précédents.

Opérateurs de préférences. Les préférences sont aussi exprimées en utilisant des formules de préférences [Cho02]. Ces formules sont utilisées par l'opérateur WINNOW pour trouver les meilleurs tuples. En d'autre termes, deux tuples peuvent apparaître en résultat s'ils ont les mêmes valeurs ou ils sont indifférents (aucun des tuples est meilleur par rapport aux autres).

Un autre exemple d'opérateur de préférence est le SKYLINE [BKS01]. Pour l'utiliser, la syntaxe SQL est enrichie par une autre clause appelée SKYLINE. Les tuples retournés par l'opérateur SKYLINE sont ceux qui ne sont pas dominés par d'autres tuples. En effet, un tuple t_1 domine un autre tuple t_2 si t_1 est aussi bon que t_2 dans toutes les dimensions et il est meilleur au moins dans une dimension.

Une autre approche qui permet de définir des critères de sélection hiérarchiques est présentée dans [LL87]. Dans cette approche, les sélections qui ont même importance sont introduites en utilisant la clause PREFER tandis que pour les moins importantes la clause FROM WHICH PREFER est utilisée. Dans ce cas, les critère de sélection introduits par les clauses de préférences sont considérées comme des filtres qui sont appliquées aux résultats de la requête sans ces critères. Lorsqu'on applique les critères de préférences, l'ordre établi par la hiérarchie est respectée. Chaque niveau de hiérarchie contient les sélections qui ont la même importance. Lorsqu'on applique les critères d'un niveau donné, seulement les résultats satisfaisant le nombre le plus élevé de critères sont gardés pour un filtrage plus loin. Si on n'a aucun résultat

lorsqu'on applique la sélection d'un niveau donné, l'opération est non validée et les niveaux suivants sont considérés.

2.4.2.2 Recommandation dans les bases de données

Dans le domaine des bases de données, les travaux de recommandation consistent à recommander souvent des requêtes en se basant sur l'historique des manipulations réalisées par l'utilisateur ou un groupe d'utilisateurs. La recommandation peut concerner des requêtes anticipées ou des requêtes alternatives susceptibles d'intéresser l'utilisateur.

Avec les résultats de chaque requête, [SDP09] proposent des recommandations des résultats supplémentaires qui sont appelés résultats YMAL (You May Also Like : "vous aimerez peut-être aussi"). Ces résultats supplémentaires recommandés pourraient être de potentiels intérêts de l'utilisateur. En fait, l'utilité d'une requête de l'utilisateur est égale au nombre de fois que l'utilisateur a posé cette même requête. Ce système fonctionne avec des approches basées sur l'historique, qui exploitent le log de requêtes pour calculer les similarités entre les utilisateurs et entre les requêtes. Les recommandations générées peuvent être soit des résultats YMAL fondés sur une requête (comme les recommandations basées sur le contenu), soit des résultats YMAL utilisateur (similaires aux recommandations de collaboration). Une autre approche de recommandation de requêtes est présentée dans [YPS09]. Son objectif est de créer automatiquement des recommandations de requêtes de jointures en se basant sur des spécifications d'input et d'output. Cette approche analyse le log de requêtes et extrait les jointures dans les requêtes précédentes. Ainsi, ces jointures seront utilisées pour générer des recommandation à l'utilisateur courant en se basant sur une approche collaborative. Toutes les requêtes de log sont utilisées pour créer un graphe de schéma général où les nœuds représentent les tables et les arcs sont les jointures entre les tables. Une requête dans le log de requêtes correspond à un sous-graphe connexe dans le graphe de schéma. A partir des nœuds qui représentent des tables figurant dans la clause WHERE et à travers les jointures avec la table intermédiaire, on peut atteindre les tables figurant dans la clause SELECT. Ainsi, aucune information sur l'utilisateur est pris en compte. L'information utilisée est extraite de la requête à savoir les jointures. Après avoir présenté le modèle conceptuel et son instanciation dans [CEP09], les auteurs présentent leur système baptisé *QueRIE* de recommandation de requêtes personnalisées (requêtes SQL complètes). Pour générer des recommandations, ce système construit d'abord le résumé pour chaque utilisateur, puis génère un résumé "prévu" (predicted summary) pour les utilisateurs et génère finalement des recommandations de requêtes basées sur le résumé "prévu". L'instanciation proposée de ce système est basée sur le filtrage collaboratif. En se basant sur l'analyse de log de requêtes, *Snipsuggest* assiste les l'utilisateur dans la composition de requêtes complexes par la recommandation d'un ensemble des additions à une clause bien spécifique [KKBS10]. Cette méthode se focalise sur l'analyse de log de requêtes qui implique la construction d'un graphe où les nœuds sont les ensembles de fragments apparaissant dans les requêtes. Les arêtes indiquent la priorité et prennent valeur dans l'intervalle $[0, 1]$. La requête partiellement écrite est identifiée comme un nœud dans le graphe et les recommandations sont les nœuds des successeurs. Ainsi, dans

cette approche la session requête est modélisée par un graphe et la requête n'est qu'un nœud dans ce graphe.

2.4.2.3 Synthèse

Malgré la richesse des travaux sur la personnalisation et la recommandation dans les bases de données, les approches proposées ne peuvent pas être directement appliquées aux entrepôts de données à cause des particularités de ces derniers. En effet, les modèles des bases de données sont incohérents avec les concepts de base des entrepôts de données, à savoir la modélisation multidimensionnelle.

		Koutrika et al. 2005	chimocki et al. 2008	Stefanidis et al. (2009)	Chatzopoulou et al (2009, 2011), Yang et al (2009)	Khoussiounova et al. (2010)
Rôle	Personnalisation	×	×			
	Recommandation			×	×	×
Type	Cognitif	×	×	×		
	Collaboratif			×	×	×
Profil	Préférences	×	×			
	Usage			×	×	×
	Source externe			×		
Collecte d'information	Explicite	×	×			
	Implicite			×	×	×
Formulation de préférences	Quantitative	×				
	Qualitative		×			
Moment (% interrogation)	Avant		×		×	
	Pendant					×
	Après	×	×	×	×	
Objet d'exploitation	Schéma		×		×	
	Requête					×
	Interface	×	×	×	×	
Technique utilisée		Enrichissement de requête	Opérateur de préférences	Enrichissement de requête	Théorie des graphes	Enrichissement de requête

Tableau 2.1 – Prise en compte de l'utilisateur dans les bases de données

Pour affiner la lecture des différents travaux portant sur la prise en compte de

l'utilisateur dans les bases de données, nous proposons une grille d'analyse basée sur les critères de comparaison que nous avons présentés dans la Section 2.4.1.

2.4.3 L'utilisateur dans les entrepôts de données

Ainsi que les bases de données, il existe deux catégories de travaux de prise en compte de l'utilisateur dans les entrepôts de données. La première catégorie concerne la personnalisation où les travaux visent à adapter le système OLAP en fonction de préférences utilisateur explicitement ou implicitement collectées. La deuxième catégorie est orientée plutôt recommandation où les travaux consistent à recommander souvent des requêtes en se basant sur les préférences ou l'historique de navigation de l'utilisateur ou d'un groupe d'utilisateurs.

2.4.3.1 Personnalisation dans les entrepôts de données

La personnalisation peut se situer à plusieurs niveaux du système OLAP. Elle peut porter sur le schéma de l'entrepôt de données, la visualisation des données (ou interface) et/ou sur l'interrogation. La personnalisation au niveau conceptuel impacte le schéma OLAP tandis que la personnalisation de la présentation impacte la manière dont sont restituées les données. Enfin, la personnalisation de l'interrogation consiste à modifier (généralement enrichir) la requête posée par un utilisateur.

2.4.3.1.1 Niveau conceptuel

L'entrepôt de données peut être personnalisé au niveau de son schéma conceptuel [GPMT09]. Ainsi, l'utilisateur de l'entrepôt de données est capable de travailler avec un schéma OLAP personnalisé. Dans ces travaux, Garrigos et al. traitent la personnalisation OLAP dynamique puisque c'est une tâche plus compliquée comme ça entraîne une interaction implicite ou explicite avec l'utilisateur. En effet, la personnalisation OLAP dynamique signifie qu'un cube OLAP adapté est créé durant le temps d'exécution en se basant sur les besoins et les actions de l'utilisateur. Cependant, la personnalisation statique diffère de la personnalisation dynamique par le fait que pour différents utilisateurs de l'entrepôt de données, les cubes OLAP sont créés durant le temps de modélisation. Par conséquent, les auteurs utilisent un langage PRML (Personalization Rules Modeling Language décrit [GG06]) pour la spécification des règles de personnalisation OLAP. Ce langage est basé sur les règles ECA(Event-Condition-Action [TSM01]).

Par ailleurs, la personnalisation est vue comme une intégration de nouveaux besoins de l'utilisateur au niveau de l'entrepôt de données [BFB08]. De ce fait, la solution qui a été exploitée est l'évolution de schémas pour intégrer ces nouveaux besoins. Cette approche permet la création de nouveaux niveaux de granularité supplémentaires dans les hiérarchies de dimension existantes de l'entrepôt offrant ainsi à l'utilisateur de nouvelles possibilités d'analyse.

Citons aussi les travaux de Golfarelli et al. qui présentent une approche de personnalisation de schéma OLAP par les constructeurs de préférence *Preference Constructors* (PC). Une algèbre qui permet de formuler les préférences des attributs, des mesures et des hiérarchies est définie en [GR09]. Une importante caractéristique de

cette algèbre proposée est la possibilité d'exprimer les préférences des attributs de hiérarchie des ensembles de *group-by* ce qui peut mener par conséquent à exprimer les préférences des faits. La fonction *Rollup* est utilisée pour étendre les préférences appliquées aux attributs sur l'hiérarchie. Les préférences peuvent être définies pour les attributs et aussi les mesures par exemple les attributs catégoriques ou numériques. Les utilisateurs peuvent exprimer leurs préférences pour les requêtes OLAP [GR09]. Dans ce cas, le problème de perdre le temps de traitement des opérations OLAP pour trouver l'information nécessaire peut être considérablement amélioré.

2.4.3.1.2 Niveau visuel

Un autre aspect de la personnalisation OLAP est la représentation visuelle des données. Mansmann et a. introduit des techniques multiples de présentation et visualisation qui peuvent être utilisés pour les différentes tâches d'analyse [MS07]. La personnalisation au niveau visuel consiste à adapter les structures de visualisation en fonction des préférences utilisateur ou des contraintes externes tels que le type et la taille du dispositif d'accès.

Dans ce contexte, certains chercheurs présentent une interface utilisateur pour les analyses OLAP où l'utilisateur est explicitement impliqué [JRTZ09a, RT08]. Par conséquent l'utilisateur possèdent une interface pour formuler les requêtes en utilisant la manipulation par un schéma graphique OLAP et des règles.

En outre, les préférences OLAP sont prises en compte avec une contrainte de visualisation pour résoudre les limitations imposés par les différents dispositifs d'accès des utilisateurs [BGM+05].

2.4.3.1.3 Personnalisation au niveau de l'interrogation

Dans le contexte des bases de données et des entrepôts de données, l'axe le plus émergent est la personnalisation de requêtes. L'idée principale derrière ce processus de personnalisation consiste à considérer les préférences utilisateur lors de la réponse à la requête [KI05]. En effet, la personnalisation de requête est considérée comme un enrichissement de la requête ou carrément une réécriture de cette requête.

Enrichissement de requête. Les méthodes d'enrichissement des requêtes supposent l'existence d'un profil de l'utilisateur. Des éléments du profil sont utilisés pour étendre la requête de l'utilisateur. Ceci se traduit par l'ajout de conditions de sélection [BGM+05] ou d'attributs d'agrégation [RTTZ07]. La version étendue de la requête est ensuite exécutée en substitution de la requête initiale afin de générer un résultat adapté à l'utilisateur.

Pour [BGM+05], les préférences utilisateur sont définies par un pré-ordre total entre les ensembles de membres de toutes les dimensions. Ainsi, cette approche est similaire à celle de Koutrika et al. [KI04b], puisque un ordre total entre les membres peut être défini à partir de degrés des intérêts (un membre ou une valeur d'attribut est préféré(e) à un autre membre si son degré d'intérêt est plus important). Ils ont aussi introduit des contraintes de visualisation qui permettent à l'utilisateur de spécifier les limitations imposées par le d'accès pour afficher les réponses à ses requêtes. Dans

ce cas, le profil utilisateur contient les préférences de l'utilisateur, les membres classés de toutes les dimensions et les contraintes de visualisation qui contrôlent la forme de visualisation des résultats de la requête

Par ailleurs, en se basant sur la requête de l'utilisateur, le schéma de l'entrepôt de données et un profil de l'utilisateur comportant un ensemble de règles, Ravat et al génèrent une requête étendue enrichie par ces règles. En effet, une règle détermine les attributs de dimension à afficher. Ainsi, les règles avec un score supérieur à un seuil sont utilisées pour enrichir la requête [RTTZ07].

Opérateurs spécifiques. Inspirés des travaux de [KK02] en bases de données, les travaux de Golfarelli et al. [GR09] se basent sur le modèle BMO ("Best Matches Only") d'exécution de requêtes où seulement les tuples du résultat qui ne sont pas dominés par d'autres sont renvoyés. Ils traitent la problématique des résultats volumineux ou vides des requêtes OLAP. Ils personnalisent la requête après son exécution. En fait, ils exploitent les préférences afin de déterminer les tuples du résultat qui sont meilleurs que tous les autres. Pour cela, Golfarelli et Rizzi ont défini une algèbre de formulation de préférences qualitatives simples (POS, NEG, CONTAIN, ...) [GR09]. Cette algèbre est enrichie par deux opérateurs de composition de préférences : Pareto ($\otimes$) pour indiquer le même degré d'importance entre deux préférences et Priorisation ($\triangleright$) pour définir un ordre de priorité entre préférences. Ces différents opérateurs sont ensuite intégrés dans une requête MDX à l'aide de clause PREFERRING.

2.4.3.2 Recommandation dans les entrepôts de données

Dans les entrepôts de données, l'utilisateur navigue interactivement dans un cube en lançant une séquence de requêtes. Ce processus est souvent pénible puisque l'utilisateur peut ne pas avoir d'idée sur ce que pourrait être la prochaine requête [Sar00]. Ainsi, en s'inspirant des travaux de recommandation dans le Web

A part les travaux de [BF09] qui recommandent des axes d'analyse, la majorité des approches de recommandation dans les entrepôts de données se focalisent sur la recommandation de requêtes afin de prendre en compte l'aspect navigationnel de l'interrogation des données OLAP. Elles visent à guider l'utilisateur lors de son exploration des données.

Nous pouvons identifier deux catégories de travaux de recommandation dans les entrepôts de données : (1) les travaux exploitant le profil utilisateur et(2) les travaux utilisant les usages ou plus particulièrement le log de requêtes.

2.4.3.2.1 Exploitation des profils utilisateurs

Certaines approches exploitent les profils et les préférences des utilisateurs [BF09, JRTZ09a, GRB11].

Bentayeb et al. proposent un système de recommandation basé sur la définition d'un nouveau opérateur OLAP appelé RoK (Roll-up with K-means) [BF09]. Cet opérateur permet de créer un nouveau niveau de granularité dans une hiérarchie de dimension en se basant sur une combinaison entre la méthode de classification K-means et un opérateur classique rollup. Dans ces travaux, les préférences utilisateur sont prises en compte pour lui recommander des axes d'analyse plus pertinents.

Citons aussi la proposition de Jerbi et al. qui supposent qu'un profil utilisateur est fourni avec la session courante [JRTZ09a]. Ils se sont inspirés des techniques de filtrage d'information en fonction de profil utilisateur pour affiner des requêtes en y ajoutant des prédicats [KI05]. L'objectif de ces travaux est de pouvoir fournir à l'utilisateur un résultat focalisé sur son centre d'intérêt, tout en exploitant des ordres (représentation qualitative des préférences). Ces derniers ne sont pas exprimés de façon absolue, mais par rapport à un contexte d'analyse donné. Ceci permet de prendre en compte le fait que les préférences peuvent varier d'un contexte d'analyse à l'autre.

Par ailleurs, inspiré des travaux de [KK02] et de [GR09], Golfarelli et al. proposent une approche où les préférences sont utilisées pour annoter la requête. Ils définissent une algèbre qui permet de formuler les préférences des attributs, des mesures et des hiérarchies [GRB11]. Une importante caractéristique de cette algèbre proposée est la possibilité d'exprimer les préférences des attributs de hiérarchie des ensembles de *group-by* ce qui peut mener par conséquent à exprimer les préférences des faits. La technique utilisée est de personnaliser une requête en traitant une sous requête de la requête courante. Dans ce cas, la requête recommandée est la sous requête qui retourne un résultat préféré non vide.

2.4.3.2.2 Exploitation des logs des requêtes

Certains auteurs exploitent les logs des requêtes contenant des séquences de requêtes précédemment exécutées par d'autres utilisateurs sur le même cube de données [Sap00, GMN09]. En effet, Sapia prévoit la prochaine requête OLAP avec un pré-traitement des données [Sap00]. Il est à noter qu'il prévoit le prototype de la prochaine afin d'améliorer les performances du système OLAP. Généralement, cette approche n'est pas retenue dans les travaux de recommandation dans le domaine des entrepôts de données car elle s'inscrit dans une perspective de prévision.

Pour aller au-delà de l'approche proposée dans [GMN09], Giacometti et al. traitent les fichiers log mais cette fois-ci les sessions sont associées à un intérêt, et les recommandations sont les requêtes des sessions précédentes ayant le même intérêt que la session courante [GMNS09]. L'importance de ces travaux réside non pas dans la recommandation des requêtes à partir des sessions qui précèdent la session courante mais dans la recommandation à partir de toutes les sessions où l'utilisateur trouve les mêmes données inattendues comme dans la session courante. Dans cette approche les préférences utilisateurs ne sont pas prises en considération.

Par ailleurs, en exploitant toujours les logs des requêtes, Giacometti et al. proposent une approche qui permet de recommander la requête suivante durant une analyse OLAP. Une analyse est considérée comme une succession de requêtes MDX, où chaque requête est représentée par un ensemble de références. La recommandation est basée sur l'exploitation de l'historique de requêtes [GMN09].

Récemment, Aligon et al. (basé sur les travaux de [GR09] et [GRB11]) proposent une approche qui permet d'extraire des préférences à partir des fichiers log de requêtes MDX en utilisant des techniques de fouille de données. Ainsi, les préférences extraites (sous forme de règles d'association) permettent d'annoter les requêtes MDX. Par la suite, ils mesurent le similarité entre deux sessions pour recommander une session

d'analyse [AGG+15].

2.4.3.3 Synthèse

Cet aperçu des travaux de prise en compte de l'utilisateur dans les entrepôts de données que nous venons de présenter montre d'abord que plusieurs travaux ont été menés pour étudier la prise en compte des utilisateurs au sein des entrepôts de données. Les travaux sur la personnalisation des entrepôts de données ont couvert plusieurs niveaux du système OLAP (niveau conceptuel, niveau visuel, niveau d'interrogation). En ce qui concerne la recommandation dans les entrepôts de données, plusieurs travaux ont été menés sur la suggestion des requêtes. Dans ce cadre de prise en compte de l'utilisateur dans les entrepôts de données, les techniques utilisées sont très variées. Ces travaux ont consisté principalement à étendre les approches habituelles en permettant de modifier les structures multidimensionnelles et/ou les mécanismes de présentation des données par une meilleure connaissance des utilisateurs.

Les approches de personnalisation et de recommandation dans les entrepôts de données présentes dans la littérature s'intéressent particulièrement à l'intégration du profil de l'utilisateur dans le processus d'analyse. Ces approches se distinguent essentiellement par la manière de déterminer le profil de l'utilisateur. Quant à l'exploitation du profil, le choix entre la personnalisation ou la recommandation dépend essentiellement de l'objectif poursuivi.

En se basant sur les critères que nous avons défini dans la Section 2.4.1, la comparaison des travaux de prise en compte de l'utilisateur dans les entrepôts de données est récapitulée dans le Tableau 2.2.

2.5 Conclusion

Au cours de ce chapitre, nous avons présenté quelques concepts et définitions importants pour la prise en compte de l'utilisateur dans les entrepôts de données. Puis nous avons présenté un état de l'art des travaux de prise en compte de l'utilisateur que ce soit dans les bases de données ou dans les entrepôts de données. Enfin, partant de la constatation du manque de distinction entre certains concepts reliés à la prise en compte de l'utilisateur dans les entrepôts de données, nous avons présenté une classification des divers travaux qui traitent la personnalisation et la recommandation sous des angles différents.

Partant de cet état de l'art, il y a eu autant de questions soulevées que de réponses apportées par cette recherche. Néanmoins, elle a permis d'éclaircir les différents nouveaux concepts utilisés en entrepôts de données, à savoir la personnalisation et la recommandation. Ceci a été particulièrement possible en faisant le rapprochement avec les autres disciplines et surtout celle des bases de données qui s'est déjà intéressée à la prise en compte de l'utilisateur, et qui elle-même a besoin des techniques adaptées pour comprendre les besoins et les perceptions des utilisateurs.

A notre connaissance, aucune approche de personnalisation présentée dans cet état de l'art n'assiste l'utilisateur à tous les niveaux de l'interrogation et utilise les préférences des utilisateurs pour les intégrer de façon interactive dans l'entrepôt afin

de les exploiter pour des fins d'analyse. En effet, tous ces travaux se sont focalisés sur l'exploitation du profil de l'utilisateur (préférences, contexte, contraintes de visualisation) pour l'enrichissement de requêtes décisionnelles. D'autre part, les systèmes de recommandation de requêtes développés ces dernières années dans le cadre des entrepôts de données, s'intéressent plus souvent à l'anticipation de requêtes décisionnelles. Ils sont également basés soit sur le profil utilisateur, soit sur l'historique de leurs requêtes.

Nous pensons que, pour présenter à l'utilisateur des informations pertinentes, le système doit intégrer dans le processus d'entreposage les préférences des utilisateurs. Plus précisément, notre point de vue consiste à impliquer l'utilisateur à tous les niveaux d'analyse : avant, pendant ou après l'interrogation de l'entrepôt de données. Tout d'abord, le contenu de l'entrepôt de données peut être personnalisé. Ensuite, lors de la formulation de sa requête, l'utilisateur peut être assisté par des recommandations personnalisées pour l'écriture de sa requête. Enfin, une fois le cube de données est construit, l'utilisateur peut être assisté pendant sa navigation dans le cube OLAP par les recommandations des chemins de navigation les plus pertinents.

Naturellement, le cœur de nos travaux est le processus de prise en compte de l'utilisateur qui peut être sous forme de personnalisation ou de recommandation. Nous pensons que le contenu de l'entrepôt de données peut être adapté en fonction des contraintes utilisateur.

Technique utilisée		Bellatreche et al. 2005	Sarawagi et al. 2008	Bentayeb et al. (2008)	Garrigos et al. (2007)	Ravat and Teste (2008)	Jerbi et al. 2009	Giacometti et al. (2009)	Golfarelli et al. (2011)	Aligon et al. (2015)
Orientation	Personnalisation	×		×		×	×	×	×	×
	Recommandation		×		×					
Type	Cognitif	×		×		×	×	×	×	×
	Collaboratif		×		×			×		
Profil	Préférences	×		×	×	×	×		×	
	Usage							×		×
	Contexte						×			
Moment (% interrogation)	Avant	×	×	×		×		×	×	×
	Pendant				×	×				
	Après		×				×			
Objet d'exploitation	Schéma		×	×	×	×	×	×	×	
	Requête	×	×					×	×	×
	Interface	×								
Collecte d'information	Explicite	×		×	×	×		×	×	×
	Implicite									
Formulation de préférences	Quantitative					×				
	Qualitative	×	×	×			×			×
Relation d'ordre		×	×		×					×
Entropie maximale			×							
Fouille de données				×						×
Règles ECA					×					
Règles ECA						×				
Théorie des graphes							×			
Distance de Hamming Distance de Levenshtein								×		
Modèle BMO Graphe de dominance									×	
Fouille de données										×

Tableau 2.2 – Prise en compte de l'utilisateur dans les entrepôts de données

Chapitre 3

Personnalisation du contenu d'un entrepôt de données

L'objectif de ce chapitre est tout d'abord de motiver la nécessité de personnaliser le contenu d'un entrepôt de données. Nous ciblons dans le contenu d'un entrepôt de données les hiérarchies de dimensions qui constituent les différents axes d'observation dans les analyses OLAP. En effet, les hiérarchies de dimensions représentent une partie substantielle du modèle d'entrepôt de données. Elles permettent aux décideurs d'examiner les données à différents niveaux de détail en utilisant les opérateurs OLAP tels que drill down et roll-up. Les niveaux de granularité qui composent une hiérarchie de dimension sont généralement fixés lors de l'étape de conception d'un entrepôt de données, en fonction des besoins d'analyse identifiées par les utilisateurs. Cependant, en réalité, les besoins des utilisateurs sont susceptibles d'évoluer et de changer dans le temps.

En empruntant le concept d'évolution de schéma issu des bases de données, nous apportons une certaine flexibilité au schéma d'un entrepôt de données pour permettre d'ajouter un nouveau niveau de hiérarchie dans une dimension. Notre objectif est d'intégrer les contraintes utilisateurs pour créer des hiérarchies de dimension nouvelles pouvant permettre des analyses OLAP non prévues par le modèle initial. Pour atteindre cet objectif, nous utilisons une technique de fouille de données permettant d'extraire des connaissances cachées à partir des données entreposées qui peuvent servir à construire de nouveaux axes d'analyse personnalisés ayant une sémantique plus riche.

Dans ce contexte, nous proposons un opérateur d'agrégation appelé *PRoCK* (Personalized Roll-up with Constrained K-means) capable de créer, grâce à la méthode des k-means contraints et au concept d'évolution de schéma, un nouveau niveau de hiérarchie personnalisé dans une dimension existante. La méthode des k-means contraints permet de découvrir de nouveaux regroupements intéressants d'un ensemble de données tout en tenant compte des contraintes utilisateur qui peut alors élaborer des analyses OLAP plus sophistiquées sur le nouveau niveau de hiérarchie créé.

3.1 Motivation

Pendant la phase de conception, la définition du modèle d'entrepôt en termes d'indicateurs, de dimensions et de hiérarchies de dimensions est établie en fonction des besoins globaux des utilisateurs.Une fois l'entrepôt de données construit, l'utilisateur ne peut réaliser que des analyses déjà prévues par ce modèle. Or, les besoins des utilisateurs peuvent évoluer dans le temps. Dans ce cas, il est important que le système décisionnel tienne compte de cette évolution.

Par ailleurs, dans le cadre d'une démarche classique d'OLAP, l'utilisateur commence par sélectionner les mesures et les dimensions qui sont susceptibles de répondre à ses besoins en termes d'analyse multidimensionnelle. Une fois le cube de données associé à ce besoin construit, l'utilisateur explore ce cube pour tenter de déceler rapidement des similarités entre les faits en fonction des différents axes d'observation. Ce sont en général les différents niveaux de granularité dans les hiérarchies de dimensions qui permettent de détecter et d'apprécier ces similarités.

Souvent, l'utilisateur s'intéresse aux faits qui concernent un groupe d'individus qu'il connaît à l'avance. Il va donc focaliser son exploration sur les zones du cube où ces individus sont présents en résumant l'information ou au contraire en l'affinant. Cependant, dans certains cas, l'utilisateur peut vouloir intégrer ses propres préférences et/ou contraintes dans le processus d'analyse en ligne afin de réaliser des analyses plus proches de ses besoins réels. Malheureusement, les systèmes OLAP actuels ne disposent pas d'outils pour tenir compte du profil utilisateur.

D'un autre côté, la fouille de données propose des méthodes de classification automatique pour regrouper les individus (au sein d'une même population) possédant des caractéristiques similaires (deux individus sont considérés comme similaires lorsque les valeurs des variables qui les décrivent sont proches). Certaines méthodes de classification peuvent même tenir compte de contraintes pour réaliser ces regroupements. Notre idée clé porte alors sur l'application des techniques de fouille avec contraintes sur les données d'un entrepôt de données afin de découvrir des regroupements de données pouvant constituer des axes d'analyse nouveaux permettant de réaliser de nouvelles requêtes décisionnelles potentiellement pertinentes pour l'utilisateur.

Dans les sections suivantes, nous montrons comment l'évolution de schéma et la fouille de données peuvent être exploitées dans le cadre de la personnalisation des axes d'analyse dans un entrepôt de données pour permettre de nouvelles analyses OLAP pertinentes.

3.1.1 Évolution de schéma dans les entrepôts de données

Traditionnellement, les modèles (schémas et données) d'entrepôts de données sont définis et fixés au moment de leur conception en fonction des besoins globaux des utilisateurs. Par ailleurs, les données d'un entrepôt de données sont non-volatiles ; la seule mise à jour possible est le rafraîchissement de l'entrepôt à partir des sources de données et ce, de façon périodique. Ainsi, lorsqu'un utilisateur exprime de nouveaux besoins d'analyse individuels, il est impossible d'y faire face avec le modèle d'entrepôt initial.

Pour faire face à ce problème, des travaux de recherche s'inspirant du concept

d'évolution de schéma issu des bases de données ont été utilisés. Une première catégorie d'approches recommande l'extension de l'algèbre multidimensionnelle afin de mettre à jour le modèle avec un ensemble d'opérateurs d'évolution de schéma [HMV99]. La deuxième catégorie, propose quant à elle des modèles de données multidimensionnelles temporelles [MV00, Hoa11]. Elle consiste à gérer différentes versions d'un entrepôt de données.

Ces deux catégories d'approches apportent une réponse à l'émergence de nouveaux besoins d'analyse qui sont engendrés par l'évolution des données, mais pas lorsqu'ils sont engendrés par des connaissances du domaine dont disposent les utilisateurs. En effet, dans les modèles existants, seules les données sont utilisées pour atteindre les objectifs d'analyse.

Ainsi, pour intégrer les nouveaux besoins individuels de l'utilisateur au sein d'un entrepôt de données, nous nous sommes inspirés des travaux sur l'évolution de schéma dans le but de pouvoir étendre les possibilités analytiques de l'entrepôt sous forme de nouveaux niveaux de hiérarchie de dimensions.

3.1.2 Couplage fouille de données/entrepôts de données

Lorsqu'un utilisateur exprime un nouveau besoin d'analyse, la première question qui se pose est comment représenter et définir ce nouveau besoin au sein de l'entrepôt ? Comme nous l'avons signalé auparavant, notre objectif est de créer de nouveaux niveaux de hiérarchies de dimensions exprimant les nouveaux besoins individuels des utilisateurs. La deuxième question qui se pose est comment obtenir ces nouveaux niveaux de hiérarchies de dimensions à partir à la fois des données déjà entreposées et des contraintes utilisateur ?

C'est pourquoi nous utilisons une méthode de fouille de données adaptée à notre problématique, à savoir la méthode des k-means contraints, pour prendre en compte à la fois les besoins de l'utilisateur et ses contraintes, tout en garantissant l'intégration structurelle (hiérarchies strictes) et sémantique (contraintes utilisateur) des données. Cette méthode permet de découvrir, à partir d'un ensemble de données, de nouveaux regroupements selon un lien sémantique défini par l'utilisateur [BK13], que nous pouvons exploiter pour créer de nouveaux niveaux de hiérarchies dans les dimensions de l'entrepôt.

Avant de présenter le principe général de notre approche de personnalisation de hiérarchies de dimensions, nous présentons en détail la méthode des k-means contraints utilisée dans notre approche.

3.2 Classification non supervisée

La classification non supervisée (aussi appelée *clustering*) est une tâche importante dans le domaine de la fouille de données. Elle consiste à regrouper les données dans des classes (ou clusters), de manière à regrouper dans un même cluster les données similaires et à séparer les données non similaires dans des clusters différents.

Étant donnée une base de données de n instances $\lambda = \lambda_1, ...\lambda_n$ d'un espace X et une mesure de dissimilarité $d(\lambda_i, \lambda_j)$ entre deux instances λ_i et λ_j, l'objectif est de regrouper les instances dans différentes classes de telle manière que la partition

obtenue optimise un critère donné. Le problème de clustering peut être donc formulé comme un problème d'optimisation.

Dans ce contexte, nous nous intéressons à la recherche d'une partition des instances λ en k classes $C_1, ..., C_k$ telle que : (1) pour tout $c \in [1, k]$, $C_c \neq \emptyset$, (2) $\bigcup_{C=1}^{k} C_c = \lambda$ et (3) pour tout $c \neq c^{'}$, $C_c \cap C_{c'} = \emptyset$.

3.2.1 Classification non supervisée sous contraintes

Depuis le début des années 2000, on tend à intégrer dans le processus de classification des contraintes afin de mieux modéliser les attentes des utilisateurs. La classification non supervisée sous contrainte est une classification dans laquelle l'utilisateur a une certaine connaissance au sujet des partitions qu'il souhaite obtenir (appelée aussi connaissance a priori) pour l'intégrer dans le processus de la classification. La principale motivation de cette méthode était la classification semi-supervisée, c'est-à-dire améliorer les techniques prédictives lorsque l'ensemble d'apprentissage ne contient que peu d'instances étiquetées. L'intuition derrière la classification sous contraintes, est que les connaissances portées par les données étiquetées peuvent guider un processus de classification non supervisée à une meilleure partition de données que celle pouvant être obtenue avec des données non-étiquetées.

Les contraintes définies par l'utilisateur sur la solution recherchée permettent d'une part de modéliser plus finement des applications réelles et d'autre part de restreindre la taille de l'espace de recherche. Ce sont des contraintes portant sur des instances ou sur des clusters. Les contraintes les plus utilisées portant sur des instances sont les contraintes "must-link" et "cannot-link", introduites pour la première fois par Wagstaff et Cardie [WC00]. Ces contraintes sont posées sur des paires d'instances individuelles.
Une contrainte must-link notée $ML(\lambda_i, \lambda_j)$ indique que deux instances λ_i et λ_j doivent être dans le même cluster :

$$\forall c \in [1, k], \lambda_i \in C_c \Leftrightarrow \lambda_j \in C_c$$

Une contrainte cannot-link notée $CL(\lambda_i, \lambda_j)$ indique que les deux instances λ_i et λ_j ne doivent pas appartenir au même cluster :

$$\forall c \in [1, k], \neg(\lambda_i \in C_c \wedge \lambda_j \in C_c)$$

Nous pouvons avoir d'autres contraintes qui sont transitivement dérivées :
- $ML(\lambda_i, \lambda_j)$ et $ML(\lambda_j, \lambda_k)$ implique $ML(\lambda_i, \lambda_k)$,
- $ML(\lambda_i, \lambda_j)$, $ML(\lambda_k, \lambda_l)$ et $CL(\lambda_i, \lambda_k)$ implique $CL(\lambda_i, \lambda_l)$ et $CL(\lambda_j, \lambda_k)$.

Algorithme des k-means contraints Plusieurs travaux sur l'intégration de contraintes dans les méthodes de classification ont vu le jour depuis l'année 2000. Ces travaux ont développé des extensions des algorithmes classiques aux contraintes must-link et cannot-link, comme par exemple COP COBWEB [WC00], COP k-means [WCRS01], classification non supervisée hiérarchique [DR05], etc. Nous détaillons dans ce qui suit l'algorithme COP k-means que nous utilisons dans le cadre de

nos travaux. En effet, COP k-means a été introduit par Wagstaff et al. pour la prise en compte de la connaissance de l'utilisateur dans la classification dynamique [WCRS01]. COP k-means est donc une extension de la méthode de k-means [Mac67] en intégrant des contraintes de l'utilisateur. Son principe consiste à contrôler la violation des contraintes dans la phase de mise à jour des classes. Nous présentons le peseudo-code de l'algorithme COP k-means (Algorithme 1) et la fonction *viloer-contraintes* (Algorithme 2).

Algorithme 1 : COP k-means (D, k, $Con_{=}$, $Con_{\neq}$)

Entrées :
- Ensemble d'apprentissage D
- Nombre de clusters k
- Ensemble des contraintes must-link $Con_{=} \subset D \times D$
- Ensemble des contraintes cannot-link $Con_{\neq} \subset D \times D$

Sortie : Ensemble des clusters $C = \{C_1, ..., C_k\}$

Début

Affecter $C_1, ..., C_k$ aux k barycentres initiaux des clusters.

répéter

Affecter chaque instance $d_i \in D$ au cluster le plus proche tel que : VIOLER-CONTRAINTES $(d_i, C_j, Con_{=}, Con_{\neq})$ = faux.

Si $C_k = \emptyset$ **alors**

sortir en échec [1]

Finsi

Pour chaque *cluster* C_i, $i \in \{1, ...k\}$ **faire**

mettre à jour son barycentre par le calcul de la moyenne de toutes les instances d_j qui lui ont été affectées

Finpour

jusqu'à *convergence*

Retourner $C = \{C_1, ..., C_k\}$

Fin

1. Les instances de données sont affectées au cluster le plus proche (qui minimise la distance entre l'instance et le barycentre) sans violer aucune contrainte. S'il n'y a pas une possibilité d'affectation sans violer une contrainte alors sortir en échec.

Algorithme 2 : Violer-contraintes $(d, C_0, Con_=, Con_{\neq})$

Entrées :
- Instance d
- Cluster C
- Ensemble des contraintes must-link $Con_= \subset D \times D$
- Ensemble des contraintes cannot-link $Con_{\neq} \subset D \times D$)

Sortie : Variable V = Vrai ou Faux

Début
  Pour chaque $(d, d_=) \in Con_=$ **faire**
    Si $d_= \neq C$ **alors**
      V= Vrai
    Finsi
  Finpour
  Pour chaque $(d, d_{\neq}) \in Con_{\neq}$ **faire**
    Si $d_{\neq} \neq C$ **alors**
      V= Vrai
    Finsi
  Finpour
  Sinon V=Faux
  Retourner V
Fin

3.3 PRoCK : un opérateur d'agrégation basé sur le k-means contraints

Nous présentons dans cette section l'opérateur d'agrégation PRoCK (Personalized Roll-up with Constrained K-means) que nous avons défini pour permettre la construction de nouveaux niveaux de hiérarchies dans une dimension existante d'un entrepôt de données.

3.3.1 Principe général

Nous avons choisi d'utiliser la méthode des k-means contraints car en plus de regrouper les instances selon un lien sémantique défini par l'utilisateur, permet de tenir compte des contraintes utilisateur. En effet, en appliquant la méthode des k-means contraints sur les instances d'un niveau de hiérarchie n d'une dimension choisi par l'utilisateur, un regroupement personnalisé des instances de n vers un niveau $n+1$ (*persn*) selon un ordre de proximité sémantique est obtenu et peut être proposé à l'utilisateur. En effet, le résultat de la classification sert à créer un nouveau niveau de hiérarchie personnalisé, *persn*, qui agrège les instances du niveau n. L'utilisateur peut alors confirmer ou infirmer la création d'un tel niveau de hiérarchie personnalisé dans l'entrepôt. Le nouveau niveau de hiérarchie offre de nouvelles possibilités d'analyse sur des faits non prévus initialement par l'entrepôt.

De façon formelle, nous définissons l'opérateur PRoCK de la manière suivante :

Définition 1. (L'opérateur PRoCK) Soient un entier positif k, une population $\Lambda = \{\lambda_1, \lambda_2, ..., \lambda_n\}$ composée de n instances, un ensemble de k classes $C = \{C_1, ..., C_k\}$, un ensemble de contraintes must-link $Cons_=$ et un ensemble des contraintes cannot-link $Cons_{\neq}$.

L'opérateur $PRoCK(\Lambda, k, Cons_=, Cons_{\neq})$ calcule l'ensemble $C = \{c_1, ..., c_k | \forall i = 1..k, c_i = barycentre(C_i)\}$ et renvoie la fonction roll-up f^c_λ telle que :

$f^c_\lambda = \{(\lambda_i \rightarrow C_j) | \forall i = 1..n$ et $\forall m = 1..k$, $dist(\lambda_i, c_j) \leq dist(\lambda_i, c_m)$ et $violer - contraintes(\lambda_i, c_j) = Faux\}$.

Exemple. Soient $X = \{x_1 = 2, x_2 = 4, x_3 = 6, x_4 = 20, x_5 = 26\}$, $C = \{C_1, C_2\}$ et $Cons_= = \{(x_2, x_4)\}$ et $Cons_{\neq} = \{x_2, x_3\}\}$.

PRoCK(X, 2, Cons) retourne l'ensemble $C = \{C_1 = 8.76, C_2 = 16\}$ ainsi que l'application : $f^c_x = \{(x_1 \rightarrow C_1), (x_2 \rightarrow C_1), (x_4 \rightarrow C_1), (x_3 \rightarrow C_2), (x_5 \rightarrow C_2)\}$

Algorithme 3 : PRoCK($\lambda_l, k, Cons$)

Entrées :
- Ensemble d'apprentissage λ_l
- Entier positif $k \geq 2$ (k = nombre de classes)
- Ensemble des contraintes $Cons$ (contraintes must-link et cannot-link)

Sortie :
- Ensemble de clusters C
- Fonction de correspondance f^C_λ

Début
  $C \leftarrow$ COP k-means($\Lambda_l, k, cons$)
  $f^c_\lambda = \{(\lambda_i \rightarrow C_j)\}$
  Retourner C
  Retourner f^c_λ
Fin

L'opérateur *PRoCK* génère automatiquement la nouvelle fonction roll-up f^c_λ. Il est ainsi plus qu'un opérateur conceptuel puisqu'il ne traite pas seulement la structure de la hiérarchies mais aussi les données de cette hiérarchie et tient compte plus particulièrement des contraintes utilisateur.

3.3.2 Ajout d'un nouveau niveau d'analyse

Nous présentons dans cette section la procédure d'ajout d'un nouveau niveau d'analyse (Algorithme 4) dans une hiérarchie de dimension d'un entrepôt de données. La première étape consiste à générer une population d'apprentissage Λ_l à partir des instances du niveau d'analyse pré-existant l. La deuxième étape consiste à classifier la population Λ_l sous les contraintes utilisateur. Cette étape est assurée par notre opérateur PRoCK (Algorithme 3). Enfin, selon le choix de l'utilisateur, la dernière étape consiste à matérialiser ou non le nouveau niveau d'analyse obtenu dans l'entrepôt de données.

Algorithme 4 : *Procédure Ajouter_ niveau* $(D, l, pl, Cons, Choix)$

Entrées :
- Dimension $D = (L, \preceq)$, un niveau $l \in L$,
- Niveau d'analyse $pl \notin L$,
- Entier positif $k \geq 2$ (k = nombre de classes)
- Ensemble des contraintes $Cons$ (contraintes must-link et cannot-link) et
- Choix de l'utilisateur Vrai ou Faux

Sortie : Dimension personnalisée D'

Début

Construction de la population d'apprentissage λ_l
$f_l^{pl} \leftarrow PRoCK(\Lambda_l, k, cons)$
Si *Choix=Vrai* **alors**
D' = Matérialiser(D, l, pl, f_l^{pl})
Finsi
Retourner D'

Fin

La dernière étape de notre approche consiste à matérialiser le nouveau niveau d'analyse pl dans le modèle d'entrepôt de données. Cette matérialisation est effectuée après validation de l'utilisateur. Pour effectuer cette opération, la procédure *Ajout_ Niveau* (Algorithme 4) exécute la fonction *Matérialiser* (Algorithme 5) sur la dimension D, à partir du niveau l en utilisant la fonction roll-up f_l^{pl} générée au cours de l'étape précédente.

Algorithme 5 : Fonction Matérialiser (D, l, pl, f_l^{pl})

Entrées :
- Dimension $D = (L = \{l_{bottom}, ..., l, ..., all\}$
- Niveau $l \in L$
- Nouveau niveau $pl \notin L$
- Fonction f_l^{pl} qui associe chaque instance de l à une instance de pl

Sortie : Dimension personnalisée D'

Début

$L' = L \bigcup \{pl\}$
$\preceq' = \preceq \bigcup_{f_l^{pl}} \{(l \rightarrow pl), (pl \rightarrow All)\}$
$D' = (L', \preceq')$
Retourner D'

Fin

Nous pouvons définir formellement la fonction f_l^{pl} de la manière suivante : étant donné une dimension $D = (L = \{l_{bottom}, ..., l, ..., all\}, \preceq)$, deux niveaux $l \in L$ et $pl \notin L$ et une fonction $f_l^{pl} : instanceSet(l) \longrightarrow dom(pl)$, Matérialiser (D, l, pl, f_l^{pl}) est une fonction qui renvoie une dimension personnalisée $D' = (L', \preceq')$ où :
$L' = L \cup \{pl\}$ et $\preceq' = \preceq \cup \{(l \rightarrow pl), (pl \rightarrow All)\}$ conformément à la fonction de

correspondance f_l^{pl}.

3.3.3 Recommandation collaborative d'axes d'analyse personnalisés

Dans ce chapitre, notre objectif est de faire évoluer l'OLAP vers d'autres possibilités d'analyse plus proches des besoins réels de l'utilisateur en intégrant l'aspect collaboratif. Dans ce cadre, nous exploitons les usages d'un utilisateur donné, pour que les autres utilisateurs similaires puissent en tirer profit, dans l'esprit d'un système collaboratif. En d'autres termes, il s'agit d'un partage des possibilités d'analyse créées individuellement. C'est ainsi qu'une fois nous avons matérialisé le nouveau niveau d'analyse personnalisé, nous pouvons le recommander à d'autres utilisateurs partageant le même datamart et plus précisément la même table de dimension. En effet, le nouveau niveau, porteur d'une sémantique peut être pertinent aux autres utilisateurs.

3.3.4 Exemple illustratif

Soit l'entrepôt de données "Foodmart" d'une organisation de ventes des produits alimentaires et d'autres produits dans des magasins aux États Unis, Canada et Mexique. Tout d'abord, nous nous intéressons au fait *sales_fact_1998* qui est relié à 4 tables de dimensions *Customer*, *Product*, *Time_by_day* et *Promotion* comme illustré dans la Figure 3.1.

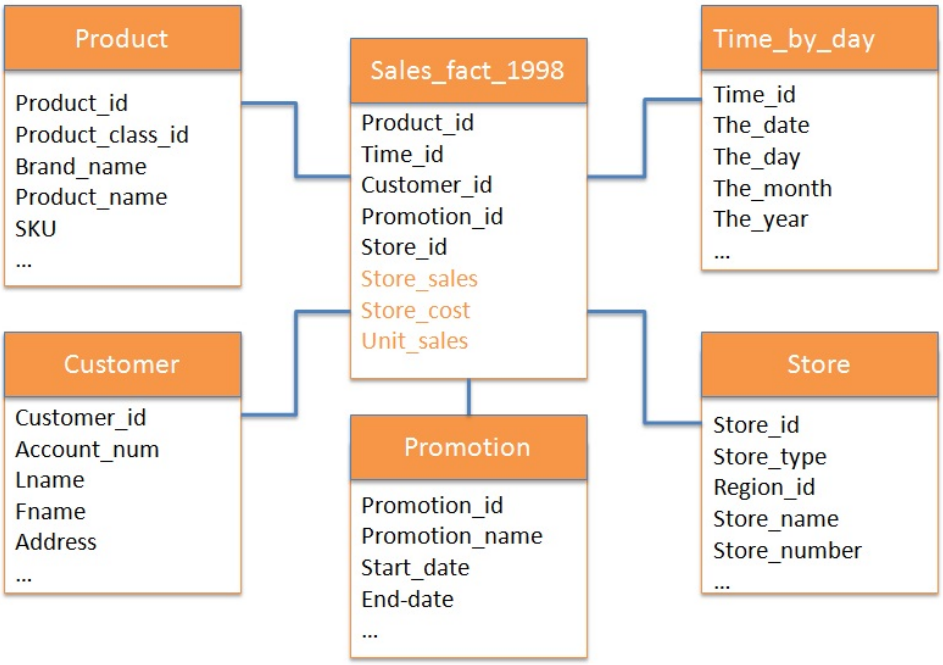


FIGURE 3.1 – Extrait du schéma de l'entrepôt de données "Foodmart"

Nous pouvons utiliser l'entrepôt de données *Foodmart* pour analyser les opérations de ventes selon plusieurs axes d'analyse (promotion, temps, magasin, produit et client). L'information sur les produits est collectée dans la dimension *Product*, tandis que les informations reliées au temps et dates sont sauvegardées dans la dimension time_by_day. Les informations concernant les clients (noms et adresses) sont sauvegardées dans la dimension *Customer*. Les attributs *Education*, *Gender*, *Marital_Status*, *Occupation* et *Yearly_Income* fournissent des informations additionnelles sur les clients. De plus, l'information sur les magasins d'alimentation individuels est collectée dans la dimension *Store*. Elle inclut les attributs *store_location*, *name*, *manager*, *size*, et *store_type* ainsi que les informations concernant les promotions sont sauvegardées dans la dimension *Promotion*.

La dimension *Time_by_day* est hiérarchisée selon deux niveaux : *Time_by_day* et *Days*. De même, la dimension *Product* est hiérarchisée selon deux niveaux : *Product* et *Product_class*.

Supposons qu'un directeur marketing veuille regrouper les produits en 3 classes selon leurs poids tout en imposant certaines contraintes. Par exemple, il veut regrouper les produits de la marque *Washington* avec la marque *Denny* mais ne veut pas avoir les produits de la marque *Red Spade* avec des produits de la marque *Club* dans le même groupe. Dans ce cas, nous pouvons utiliser le clustering contraint afin de prendre en compte les contraintes de l'utilisateur.

Pour améliorer la qualité de son analyse, il peut alors ressentir le besoin d'ajouter un nouveau niveau d'analyse qui doit regrouper les produits selon la variable (attribut poids) *Weight* du niveau "Product". Dans ce cas, il serait intéressant de classifier directement chaque produit sur l'attribut *Weight* du niveau d'analyse "Product" tout en précisant les contraintes sur les instances pour obtenir un regroupement des produits respectant les contraintes de l'utilisateur.

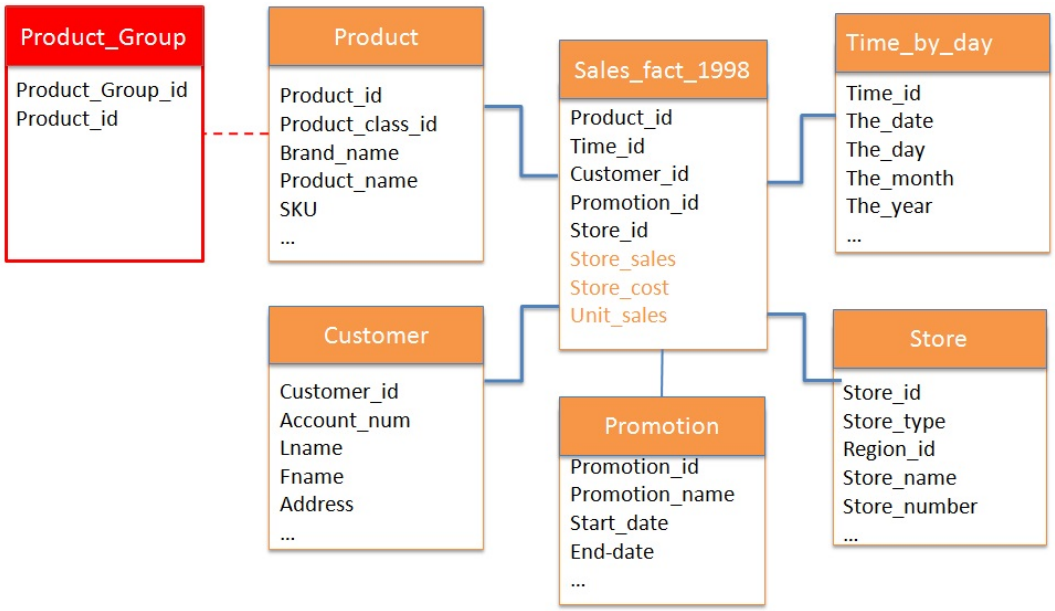


Figure 3.2 – Schéma de l'entrepôt de données Foodmart enrichi

Pour obtenir le nouveau niveau de hiérarchie *Product_ Group* à partir du niveau *Product*, nous appliquons la méthode des kmeans contraint sur les instances du niveau *Product*. Nous obtenons alors 3 clusters de produits qui vont représenter les 3 regroupements possibles dans le niveau *Product_ Group*. Si la construction de ce niveau est confirmée par l'utilisateur, ce dernier peut poser de nouvelles requêtes décisionnelles sur le niveau *Product_ Group*. De manière plus générale, la première étape consiste à construire la population d'apprentissage λ_l à partir des instances du niveau d'analyse l. La population λ_l sera décrite directement par les attributs de l. Dans ce cas, λ_l est l'ensemble des 1560 produits du niveau de hiérarchie *Product* qui sont décrits par leurs descripteurs dans l'entrepôt de données *Foodmart* (Figure 3.3).

Product_ id	*Product_ brand*	*Product_ name*	*Net_ weight*
1	Washington	Washington Berry Juice	6.39
2	Washington	Washington Mango Drink	4.42
12	Jeffers	Jeffers Oatmeal	6.89
13	Jeffers	Jeffers Corn Puffs	10.4
16	Blue Label	Blue Label Canned Beets	18.2
17	Blue label	Blue Label Creamed Corn	3.9
50	Club	Club Havarti Cheese	7.94
51	Club	Club Cheese Spread	5.4
75	Red Spade	Red SpadePotato Salad	14.8
85	Golden	Golden Frozen Puncakes	5.11
134	Denny	Denny Plastic Spoons	4.82
...			

FIGURE 3.3 – Ensemble d'apprentissage = Ensemble de produits

Nous prenons, à titre d'exemple, le paramètre k égal à 3, l'exécution de l'opérateur PRoCK sur l'entrepôt de données *Foodmart* et plus précisément sur la table de dimension *Product* nous donne l'ensemble $C = \{C_1(1.3, 7.42), C_2(7.9, 14.1), C_3(15, 20.8)\}$ ainsi que la fonction de correspondance : $f_{Product}^{Product\_Group} = \{$(Washington Berry Juice, C_1), (Washington Mango Drink, C_1), (Jeffers Oatmeal, C_1), (Jeffers Corn Puffs, C_2), (Jeffers Corn Puffs, C_2), (Blue Label Canned Beets, C_3), (Blue Label Creamed Corn, C_1) , (Club Havarti Cheese, C_2), (Club Cheese Spread, C_1), (Red SpadePotato Salad, C_3), (Golden Frozen Puncakes, C_1), (Denny Plastic Spoons, C_1), ... $\}$.

Product_ id	*Product_ brand*	*Product_ name*	*Net_ weight*	Cluster
1	Washington	Washington Berry Juice	6.39	1
2	Washington	Washington Mango Drink	4.42	1
12	Jeffers	Jeffers Oatmeal	6.89	1
13	Jeffers	Jeffers Corn Puffs	10.4	2
16	Blue Label	Blue Label Canned Beets	18.2	3
17	Blue label	Blue Label Creamed Corn	3.9	1
50	Club	Club Havarti Cheese	7.94	2
51	Club	Club Cheese Spread	5.4	1
75	Red Spade	Red SpadePotato Salad	14.8	3
85	Golden	Golden Frozen Puncakes	5.11	1
134	Denny	Denny Plastic Spoons	4.82	1
...				

FIGURE 3.4 – Ensemble de produits classifié

A la fin de l'application de la méthode de classification sur les instances du niveau de hiérarchie *Product*, il est alors possible de créer à partir de ce niveau, un nouveau niveau de hiérarchie *Product_ Group*, directement lié au niveau *Product* par le lien d'agrégation (*Product* → *Product_ Group*).

La dernière étape consiste à matérialiser le nouveau niveau d'analyse *Product_ Group* au coeur du schéma de l'entrepôt de données *Foodmart* grâce à la fonction *Matérialiser (D, l, pl,* f_l^{pl}*)*. Dans notre cas, la création du nouveau niveau de hiérarchie *"Product_ Group"* au dessus du niveau *"Product"* consiste à exécuter la fonction : $Matérialiser(Product, Product, Product\_Group, f_{Product}^{Product\_Group})$.

Classe	Intervalle	Nombre de produits
C_1	$[1.3 - 7.42]$	362
C_2	$[7.5 - 14.1]$	646
C_3	$[14.7 - 20.8]$	552

Tableau 3.1 – Nouveau niveau hiérarchique personnalisé *Product_ Group*

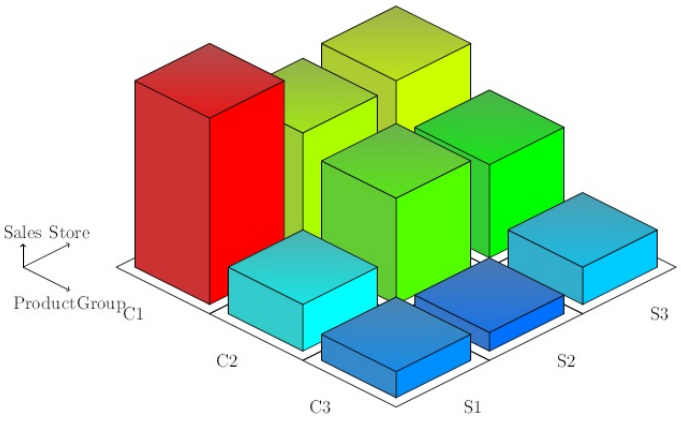


Figure 3.5 – Cube OLAP en fonction de *Product_ Group*, *Store* et *Time*

Par conséquent, notre système extrait des connaissances automatiquement à partir de l'entrepôt de données pour fournir des classes de produits potentiellement pertinentes pouvant s'agréger en un nouveau niveau plus grossier *Product_ Group*. L'utilisateur peut soumettre alors des requêtes OLAP plus élaborées sur le niveau *Product_ Group*.

Les résultats que nous avons obtenus mettent en évidence les points suivants : Le nouvel axe d'analyse créé permet d'étudier efficacement l'impact du poids des produits sur les ventes. L'utilisation de ce nouvel axe dans l'analyse OLAP montre une corrélation assez forte entre le niveau des ventes et le prix des produits (Figure 3.5). Ce qui permet d'aider le directeur de marketing à prendre des décisions plus adéquates il peut par exemple fournir les magasin par des produits plutôt de la classe 1 (produits légers) qui coûtent moins cher au niveau de la livraison mais ils ont plus

de revenus que les produits de la classe 2 et 3.

3.4 Application de PRoCK : Impact de l'utilisation d'Internet sur le développement des pays africains

Nous présentons dans cette section un deuxième exemple pour montrer l'intérêt de PRoCK dans les systèmes décisionnels. Cela concerne l'analyse de l'impact d'Internet sur le développement des pays africains. En effet, Internet est un nouveau vecteur de développement et de commerce. Nous pouvons évaluer ce nouvel indicateur de développement pour chaque pays en mesurant le nombre d'utilisateurs d'Internet par rapport à la population globale de chaque pays. La Figure 3.6 indique le nombre des utilisateurs qui accèdent à Internet au sein d'un pays. Ce graphique représente les données issues de "The World Factbook"[2]. Il contient le nombre des utilisateurs d'Internet et la population de 9 pays africains. Les statistiques varient d'un pays à un autre et le Nigeria occupe une place assez exceptionnelle comme étant le pays le plus peuplé d'Afrique.

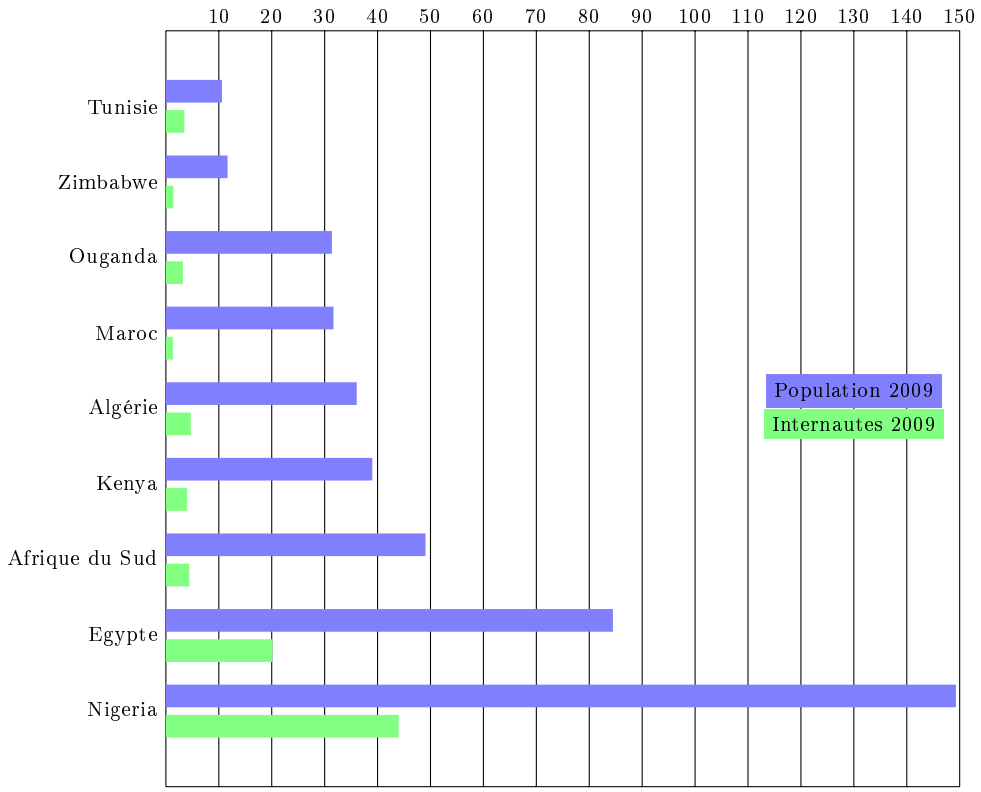


FIGURE 3.6 – Utilisateurs d'Internet en Afrique

2. https ://www.cia.gov/library/publications/the-world-factbook/rankorder/2153rank.html

Supposons que l'un des objectifs d'analyse est de savoir si le pays est développé ou non par l'impact de l'utilisation d'Internet. Pour trouver une réponse à cette question, nous avons essayé d'explorer l'utilisation de l'Internet à travers la dimension "*Pays*" la hiérarchie est organisée comme illustré dans la Figure 3.7. Pour une analyse plus ciblée, l'utilisateur peut alors ressentir le besoin d'ajouter un nouveau niveau d'analyse *PaysGroup* qui doit regrouper les pays en fonction du taux d'utilisation d'Internet. Pour atteindre cet objectif, notre idée consiste à extraire automatiquement des connaissances à partir de l'entrepôt de données pour fournir éventuellement des clusters pertinents des pays.

Figure 3.7 – Schéma de la dimension Pays

Dans ce cas, il serait intéressant de décrire directement chaque pays par les attributs *population* et *nombre des utilisateurs d'Internet*. Le système est alors en charge de regrouper les pays en fonction de ces nouvelles informations et de créer à l'utilisateur un niveau de granularité sémantiquement plus riche *PaysGroup* pour des requêtes OLAP plus élaborées.

3.4.1 Application de PRoCK

Chaque utilisateur peut vouloir un regroupement spécifique des données. Par conséquent, une façon de trouver le meilleur regroupement pour chaque utilisateur est d'intégrer les préférences de l'utilisateur. Ces dernières sont présentées comme des contraintes must-link et cannot-link entre les paires des instances de données. L'opérateur peut alors invoquer la méthode de clustering contraint *COP k-means* pour classifier automatiquement les pays selon ces deux attributs. Nous allons étudier un scénario d'analyse avec 3 cas de contraintes différentes.

1^{er} cas : Soit Λ = {Tunisie, Kenya, Zimbabwe, Algérie, Ouganda, Maroc, Égypte, Nigeria, Sud d'Afrique}. En fixant k=3 et sans l'application des contraintes, nous pouvons obtenir les clusters C_1, C_2 et C_3 comme illustré dans le tableau 3.4.1.

$2^{ème}$ cas : L'utilisateur peut vouloir trouver les pays du nord de l'Afrique notamment Égypte, Maroc, Tunisie et Algérie dans le même groupe. Ainsi, il peut introduire les contraintes suivantes :

- Must-link(Maroc,Égypte)
- Must-link(Maroc,Tunisie)
- Must-link(Maroc,Algérie)

Par conséquent, Λ = {Tunisie, Kenya, Zimbabwe, Algérie, Ouganda, Maroc, Égypte, Nigeria, Sud d'Afrique} et $Cons_{=}$ = {(Maroc, Égypte), (Maroc, Tunisie),(Maroc, Algérie)} et $Cons_{\neq} = \emptyset$.

PRoCK$(\Lambda, 3, Cons)$ retourne l'ensemble C = {c_1 = { Égypte, Maroc, Algérie, Nigeria, Tunisie }, c_2 = { Ouganda, Sud d'Afrique, Kenya }, c_3 = { Zimbabwe } }.

Pays	Utilisateurs(2009)	Population	Cluster
Tunisie	3 500 000	10 589 025	C_3
Zimbabwe	1 423 000	11 651 858	C_3
Ouganda	3 200 000	31 367 972	C_3
Maroc	13 213 000	31 671 474	C_1
Algérie	4 700 000	36 057 838	C_3
Kenya	3 996 000	39 002 772	C_3
Sud d'Afrique	4 420 000	49 052 489	C_3
Égypte	20 136 000	84 474 000	C_1
Nigeria	43 989 000	149 283 240	C_2

Tableau 3.2 – Utilisateurs Internet avec clusters

La fonction roll-up f^c_λ = {(Maroc $\rightarrow C_1$), (Algérie $\rightarrow C_1$), Nigeria $\rightarrow C_1$), Tunisie $\rightarrow C_1$), Égypte $\rightarrow C_1$), (Ouganda $\rightarrow C_2$), (Kenya $\rightarrow C_2$), (Sud d'Afrique $\rightarrow C_2$), (Zimbabwe $\rightarrow C_3$)}.

$3^{ème}$ cas : Le tableau 3.4.1 montre que le Nigeria qui s'est "invité" dans le cluster C_1, ce qui n'est pas vraiment ce qu'on souhaite quand on sait le caractère atypique de ce pays. Nous décidons donc d'introduire la contrainte suivante : cannot-link (Maroc, Nigeria). En appliquant ces contraintes, on peut obtenir le résultat souhaité.

Soit Λ = {Tunisie, Kenya, Zimbabwe, Algérie, Ouganda, Maroc, Égypte, Nigeria, Sud d'Afrique }, $Cons_=$ ={ (Maroc, Égypte), (Maroc, Tunisie),(Maroc, Algérie)} et $Cons_{\neq}$ = {(Maroc, Nigeria) }.

$PRoCK(\Lambda, 3, Cons)$ retourne l'ensemble $C = \{c_1 = \{$(Égypte, Maroc, Algérie, Tunisie}, c_2 ={ Ouganda, Sud d'Afrique, Kenya, Zimbabwe}, c_3 ={Nigeria}}.

La fonction roll-up $f^{PaysGroup}_{Pays}$ = {(Maroc $\rightarrow C_1$), (Algérie $\rightarrow C_1$), Tunisie $\rightarrow C_1$), Égypte $\rightarrow C_1$), (Ouganda $\rightarrow C_2$), (Zimbabwe $\rightarrow C_2$), (Kenya $\rightarrow C_2$), (Sud d'Afrique $\rightarrow C_2$), Nigeria $\rightarrow C_3$)}.

3.4.2 Analyses OLAP impliquant le nouveau niveau d'analyse

A la fin de la classification, notre algorithme va créer un nouveau niveau d'analyse *PaysGroup* dans la dimension *Pays*.

Pour matérialiser le nouveau niveau d'analyse "*PaysGroup*" dans la dimension "*Pays*", notre algorithme effectue la fonction Matérialiser (*Pays*, *Pays*, *PaysGroup*,$f^{PaysGroup}_{Pays}$).

Par conséquent, PRoCK fournit un moyen de faire face à la structure de la hiérarchie et de ses données par rapport aux contraintes de l'utilisateur. En fait, il génère automatiquement la nouvelle fonction roll-up basée sur les contraintes d'utilisation.

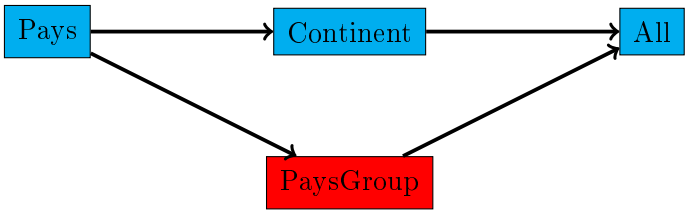


Figure 3.8 – Schéma enrichi de la dimension Pays

Un utilisateur peut faire des analyses en fonction du nouveau niveau *PaysGroup*. Ce dernier contient 3 regroupements (3 clusters) étiquetées par l'utilisateur respectivement Afrique du nord, Afrique subsaharienne et Nigeria (Tableau 3.3). Par conséquent, l'utilisateur peut soumettre de nouvelles requêtes, il peut analyser le PIB (Produit Brut Intérieur) non pas par pays (niveau de la hiérarchie *Pays*) mais par groupe de pays (nouveau niveau *PaysGroup*) comme illustré dans la Figure 3.9. Par ailleurs, nous pouvons recommander ce nouveau niveau *PaysGroup* aux autres utilisateurs appartenant à la même communauté d'utilisateurs pour qu'ils puissent en tirer profit, dans l'esprit d'un système collaboratif.

Cluster	Instances	PayGroup
1	Algérie Tunisie Égypte Maroc	Afrique du nord
2	Ouganda Zimbabwe Kenya	Afrique subsaharienne
	d'Afrique	Sud
3	Nigeria	Nigeria

Tableau 3.3 – Création du niveau d'analyse *PaysGroup* dans la dimension *Pays*

3.5 Implémentation

3.5.1 Environnement de développement

Nous avons implémenté l'opérateur *PRoCK* en PL/SQL au sein du SGBD Oracle 11g. Le résultat est un package portant le même nom *PRoCK* offrant toutes les fonctionnalités permettant de créer des hiérarchies de dimension personnalisées en tenant compte des contraintes utilisateur. Ainsi, nous avons paramétré ce package pour qu'il s'adapte à n'importe quel entrepôt de données. Nous avons conçu aussi une interface qui permet à un utilisateur de choisir un niveau de hiérarchie de dimension ainsi que ses contraintes pour appliquer ensuite l'opérateur *PRoCK* qui va créer un nouveau niveau agrégé à partir du niveau choisi par l'utilisateur.

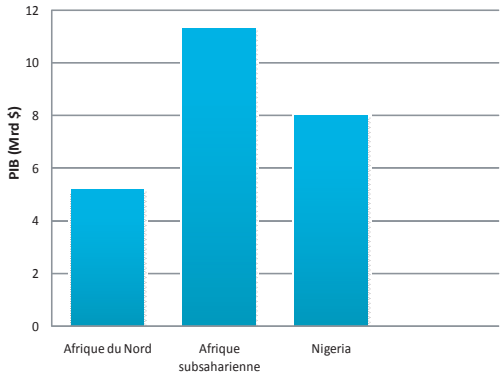


FIGURE 3.9 – Analyse du PIB en fonction de *PaysGroup*

PRoCK est basé sur un package composé de 32 fonctions PL/SQL. Le package est implémenté sur Oracle. Pour cela, il faut nécessairement se connecter, tout d'abord, au compte Oracle. Il suffit de remplir l'ensemble des informations afin de mettre en place la connexion entre le compte Oracle et l'application Java. Une fois la connexion est effectuée, l'interface illustrée dans la Figure 6.2 se présente et nous permet de suivre les étapes d'exécution du package.

3.5.2 Mise en œuvre

Pour implémenter PRoCK au sein du SGBD Oracle, nous avons utilisé 3 tables relationnelles pour stocker les ensembles de données : PRoCK_partition, PRoCK_constraint et PRoCK_means (Figure 3.11).

- *PRoCK_partition* : Cette table contient les informations décrivant les plans de partitionnement des tables. On y retrouve des informations telles que la table à partitionner, le nombre de clusters, le nom des des colonnes devant utilisées comme dimensions à agréger.
- *PRoCK_constraint* : Cette table contient l'ensemble des contraintes définies par l'utilisateur. Elle permet pour chaque plan de partitionnement de spécifier quelles sont les paires des instances ayant des contraintes must-link ou cannot-link.
- *PRoCK_means* : Cette table contient le centre initiaux initialisés aléatoirement parmi les nœuds non impliqués dans des contraintes.

Après le processus de classification, la nouvelle hiérarchie est construite en utilisant des opérateurs SQL : le nouveau niveau est créé avec la commande *create table* et la fonction *roll-up* est établie avec une association clé primaire/clé étrangère entre le nouveau niveau et le niveau existant.

Le package *PRoCK* est composé de 32 fonctions PL/SQL (Chapitre 6). Nous déroulons ici un scénario simple d'utilisation du package *PRoCK*. En effet, nous détaillons la démarche d'obtention des résultats présentés dans l'exemple cité dans la section 3.3.1.

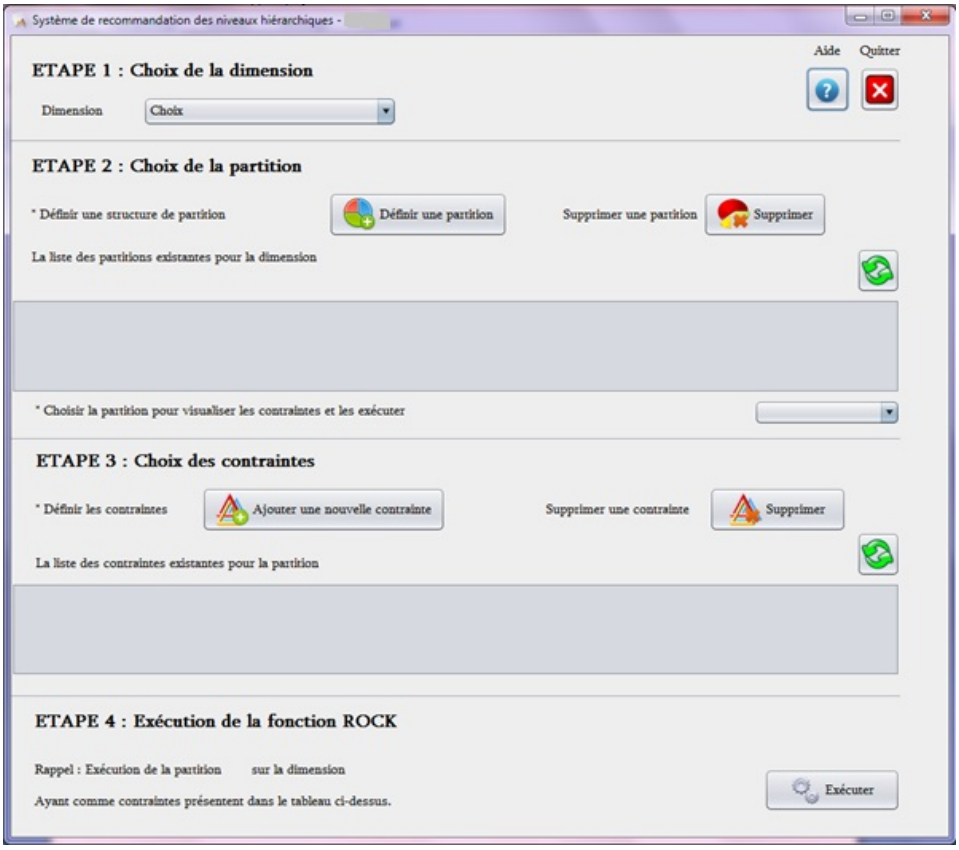


Figure 3.10 – Page de visualisation de PRoCK

Étape 1 Création de la table

create table T (id VARCHAR2(10), b NUMBER);
insert into T values ('x1',2);
insert into T values ('x2',4);
insert into T values ('x3',6);
insert into T values ('x4',20);
insert into T values ('x5',26);
commit;

Étape 2 Création du descripteur du partitionnement

Select PRoCK.create_partition_descriptor('PP1', 'Plan de partitionnement 1 de la table T','T',2,'b',null, 'Classe') from dual;

Ce descripteur sera dans la suite utilisé (via son id "PP1") pour toutes les opérations de ce partitionnement. Il a pour avantage de définir en un seul point les partitionnements que l'on souhaite obtenir à partir d'une ou plusieurs tables.

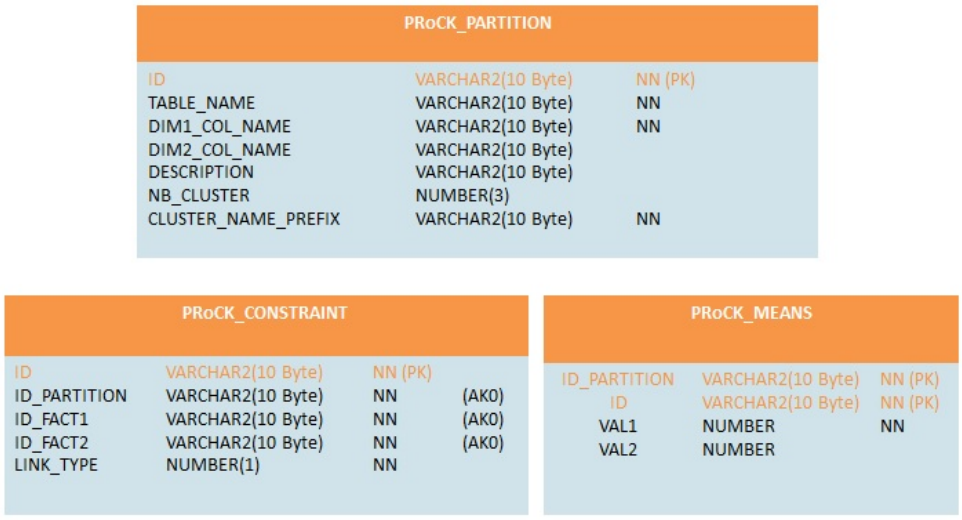


FIGURE 3.11 – Tables utilisées dans l'implémentation du *PRoCK*

Étape 3 Définition des contraintes

Select PRoCK.Add_constraint(P_id_constraint ⇒ *'PPC1',*
P_id_partition ⇒*'PP1', P_ID_fact1* ⇒ *'x2',*
P_id_fact2 ⇒ *'x4', P_link_type* ⇒ 1) *from dual;*

La fonction *Add_constraint* permet de définir une contrainte entre deux instances de données. Le paramètre *P_link_type* prend pour valeurs 1 (must-link) et 0 (cannot-link).

Étape 4 Initialisation des centroïdes initiaux

Select PRoCK.init_means_random('PP1') from dual;

Les barycentres initiaux sont choisis parmi les instances de données non impliquées dans des contraintes. Ceci permet de démarrer sans violations de contraintes.

Étape 5 Exécution de PRoCK sur le plan de partitionnement

Select PRoCK.CK('PP1') from dual;

La fonction PRoCK exécute l'algorithme du COP k-means contraints suivant les paramètres définis dans le plan de partitionnement passé en paramètre et en respectant les contraintes définies sur ce plan.

Étape 6 Affichage des résultats

Select PRoCK.print_partition_content(P_id_partition ⇒ *'PP1') from dual;*

La fonction *print_partition_content* affiche en mode console le résultat du partitionnement sous contrainte passé en paramètre. Cette fonction permet d'obtenir des résultats des traitements effectués sur des petites tables. Par exemple pour l'affichage du contenu du cluster "Classe 0" :

3.6 Conclusion

Nous avons présenté dans ce chapitre une approche de personnalisation du contenu des entrepôts de données. Cette approche vise à personnaliser le schéma de l'entrepôt de données en fonction des besoins des utilisateurs qui dépendent de leurs préférences et/ou de leurs contraintes. Plus précisément, il s'agit de faire évoluer les axes d'analyse d'un entrepôt, créés pendant la phase de conception, vers de nouvelles possibilités d'analyse grâce à la création de nouveaux niveaux de hiérarchie dans les dimensions.

Pour atteindre cet objectif, nous avons combiné la fouille de données avec la technologie des entrepôts de données pour la création d'axes d'analyse nouveaux sémantiquement plus riches et tenant compte des contraintes utilisateur.

La création d'un nouveau niveau de hiérarchie dans une dimension s'appuie sur la découverte de nouvelles structures naturelles grâce au principe d'une méthode de classification avec contraintes (k-means contraints). Nous avons alors conçu et implémenté un opérateur d'agrégation, nommé *PRoCK* (Personalized Roll-up with Constrained K-means) que nous avons intégré au sein du SGBD Oracle. Les expériences et les tests que nous avons menés attestent de l'intérêt de notre approche en fournissant des axes d'analyse pertinents à l'utilisateur.

L'approche de personnalisation que nous proposons présente plusieurs avantages. Elle est centrée utilisateur, dynamique, et permet de créer plusieurs niveaux de hiérarchies dans une même dimension, sémantiquement plus riche que les hiérarchies existantes.

Nous pensons qu'il y a d'autres pistes de recherche qui doivent être poursuivies. En particulier, il faut tenir compte des contraintes qui dépendent du contexte utilisateur tant les préférences utilisateur varient en fonction du contexte. Par exemple, les exigences d'un décideur peuvent changer d'un contexte à l'autre. Enfin, nous pensons qu'il est intéressant de combiner les concepts de personnalisation et de collaboration. L'intérêt est alors de pouvoir exploiter les connaissances d'un utilisateur donné, pour que les autres utilisateurs appartenant à la même communauté (au sein d'une même organisation par exemple) puissent en tirer profit, dans l'esprit d'un système collaboratif. En d'autres termes, il s'agit d'un partage des possibilités d'analyse créées individuellement. Ainsi, à partir d'un entrepôt initial qui constitue une base de travail, assurant l'intégrité des données, l'aspect collaboratif va se porter sur le développement, l'enrichissement incrémental de nouveaux axes d'analyse à travers la création de nouveaux niveaux de granularité définissant ou enrichissant des hiérarchies de dimension existantes.

Chapitre 4

Recommandation interactive de requêtes décisionnelles

Nous présentons dans ce chapitre une approche de recommandation interactive de requêtes décisionnelles qui s'inscrit dans la continuité de nos travaux sur la prise en compte de l'utilisateur dans les entrepôts de données. Cette approche permet d'aider l'utilisateur à formuler de nouvelles requêtes décisionnelles pour construire des cubes OLAP pertinents en s'appuyant sur les précédentes requêtes décisionnelles, ce qui lui permet d'anticiper sur ses besoins d'analyse futurs. Cette approche repose sur l'extraction des motifs fréquents à partir d'une charge de requêtes associée à un ou à un ensemble d'utilisateurs appartenant à la même communauté d'acteurs d'une organisation.

Plus précisément, nous proposons un procédé interactif d'aide à l'écriture de requêtes décisionnelles dans lequel il est proposé à l'utilisateur un choix d'attributs les plus fréquemment utilisés dans les précédentes requêtes décisionnelles. On considère ici les requêtes décisionnelles qui permettent de construire les cubes OLAP initiaux. Le choix des faits à observer en termes de mesures et de dimensions ainsi que le choix de leur granularité constituent de vraies problématiques de recherche pouvant s'apparenter au problème de la matérialisation de vues.

Notre intuition est que la pertinence d'une requête décisionnelle est fortement corrélée avec la fréquence d'utilisation par l'utilisateur (ou un ensemble d'utilisateurs) des attributs associés à l'ensemble de ses (leurs) requêtes précédentes. Notre approche de formulation de requêtes décisionnelles est collaborative puisqu'elle permet à l'utilisateur de soumettre à l'entrepôt de données des requêtes pertinentes construites, pas à pas, à partir des attributs les plus fréquemment utilisés par l'ensemble des acteurs de la communauté d'utilisateurs à laquelle il appartient.

4.1 Motivation

L'analyse en ligne commence généralement par la construction de cube données. La construction d'un cube de données à partir d'un entrepôt se fait grâce à une requête décisionnelle. En effet, le cube de données ne présente pas une structure de stockage statique pour les données de l'entrepôt de données. Il s'agit ici de définir

les différents cubes de données à extraire à partir de l'entrepôt de données pour répondre à des besoins spécifiques des utilisateurs. L'objectif est alors de pouvoir recommander des requêtes décisionnelles pertinentes à l'utilisateur. En effet, l'utilisateur peut créer plusieurs cubes de données correspondant à des besoins d'analyse différents. Cependant, le choix des requêtes correspondant aux cubes OLAP pouvant être intéressants pour l'utilisateur est une tâche fastidieuse qui peut engendrer une perte de temps. Il s'agit de choisir les faits à analyser en termes de mesures et de dimensions en choisissant le bon niveau de granularité pour permettre d'effectuer des analyses OLAP intéressantes pour l'utilisateur. Or, l'utilisateur veut souvent créer de nouveaux cubes OLAP en formulant de nouvelles requêtes décisionnelles. En effet, un même cube de données n'est pas toujours pertinent pour des tâches d'analyse différentes. Nous pensons que l'utilisateur doit être en mesure de construire les cubes de données qui sont les plus appropriés pour ses tâches d'analyse actuelles. Ceci est important car les besoins des utilisateurs peuvent changer presque quotidiennement et la construction des cubes de données doit bien sûr suivre ces changements.

Par conséquent, aider l'utilisateur à construire son cube ou ses cubes de données est une tâche importante. C'est ainsi que nous nous sommes intéressés à la recommandation de requêtes décisionnelles pour aider l'utilisateur à construire des cubes OLAP pertinents. En effet, ces dernières années, l'intérêt porté aux techniques de recommandation s'est agrandi, en raison de leur succès et leurs performances dans de nombreux domaines d'application notamment le Web [AT05]. Ces techniques ont donc retenu notre attention et plus particulièrement celles exploitant l'historique des requêtes utilisateurs.

Face à la nécessité d'offrir davantage de flexibilité pour répondre au mieux aux besoins des utilisateurs, plusieurs propositions sont apparues dans la littérature faisant appel aux outils de recommandation. Les termes de requête recommandée désignent une requête existante (ou calculée) issue d'un ensemble de requêtes posées sur l'entrepôt (fichier log de requêtes par exemple) ou des préférences utilisateur. Nous pouvons citer dans ce contexte les travaux de Giacometti et al. [GMN09] et de Jerbi et al. [JRTZ09b]. Dans les travaux de Giacometti et al., le principe est de calculer une similarité entre la séquence courante de requêtes de l'utilisateur et les séquences de requêtes précédentes [GMN09]. Par ailleurs, Jerbi et al. ont proposé une méthode de personnalisation qui prend en compte les préférences de l'utilisateur en combinant le contexte et le profil pour recommander des requêtes [JRTZ09b].

A notre connaissance, toutes les méthodes proposées s'appuient uniquement sur les requêtes utilisateurs déjà posées et exclut de ce fait l'utilisateur lui-même. Très peu de ces systèmes accompagnent le décideur durant la totalité du processus d'aide à la décision. Ils sont pour la plupart des systèmes, "intelligents" ou non, cherchant à optimiser des solutions tout en offrant peu de possibilités d'intervention à l'utilisateur pour l'aider à écrire des requêtes décisionnelles pertinentes. Nous pensons donc que les systèmes d'aide à la décision doivent impliquer davantage l'utilisateur et l'aider à formuler de nouvelles requêtes décisionnelles, ce qui lui permet d'anticiper sur ses besoins d'analyse futurs.

Ainsi, l'enjeu réside dans le fait de créer une interaction intelligente entre le système décisionnel et l'utilisateur. Pour cela, nous proposons un nouveau système de recommandation interactif, à base d'usages, dans le but d'obtenir des recommanda-

tions instantanées de requêtes décisionnelles. Pour atteindre cet objectif, nous utilisons les techniques de fouille de données et plus précisément l'extraction des motifs fréquents à partir de fichiers logs de requêtes.

4.2 Extraction des motifs fréquents

L'extraction des motifs fréquents présente sans doute une tâche intéressante qui a attiré l'attention de plusieurs chercheurs en fouille de données. En effet, le concept des motifs fréquents a été introduit par Agrawal et al. [AMS+96]. L'extraction des motifs fréquents présente une phase primordiale dans plusieurs tâches de fouille de données telles que l'extraction des règles d'association, l'extraction des corrélations, les schémas séquentiels, etc.

Par ailleurs, l'analyse du panier de la ménagère est l'une des applications typiques de l'extraction des motifs fréquents. Utilisée dans le domaine de la grande distribution, elle permet d'analyser les tickets de caisse des clients afin de comprendre leurs habitudes de consommation, agencer les rayons du magasin, organiser les promotions, gérer les stocks, etc. Dans le cadre des bases de données de ventes par exemple, un tuple consiste en une transaction regroupant l'ensemble des articles achetés appelés *items*. Ainsi une base de données est un ensemble de transactions, qu'on appelle aussi *base de transactions* ou *contexte d'extraction*.

4.2.1 Définitions

Cette section définit les principaux concepts liés à l'extraction des motifs fréquents. Nous illustrons ces concepts par la base de transactions illustrée dans la Figure 4.1. La table $\mathcal{D}$ (Figure 4.1) donne pour chaque élément, identifié par un numéro unique *tid* (transaction id), l'ensemble de ses transactions.

$\mathcal{D}$	
Tid	**Transaction**
1	a b
2	a c
3	c d
4	b c d
5	a b c d

FIGURE 4.1 – Exemple d'une base de transactions $\mathcal{D}$

Définition 2. (Item). Un item est tout élément, article, attribut, etc. appartenant à un ensemble fini d'éléments distincts $\mathcal{I} = \{x_1, x_2, ..., x_m\}$.

Exemple. Dans les applications de type analyse du panier de la ménagère, les articles en vente dans un magasin sont des items. L'ensemble $\mathcal{I}$ de l'exemple cité dans la Figure 4.1 contient les items a, b, c et d.

Définition 3. (Motif). On appelle *motif* tout sous-ensemble d'items de $\mathcal{I}$. Un motif constitué de k items sera appelé un $k-motif$.

Exemple. Le motif $\{a, b, c\}$ est un 3-motif noté *abc*.

Définition 4. (Transaction). Une transaction est un motif identifié par un identificateur unique *tid*. L'ensemble de tous les identificateurs de transactions, *tids*, sera désigné par l'ensemble $\mathcal{T}$.

Exemple. L'ensemble des produits achetés par un client dans un hypermarché représente une transaction.

Définition 5. (TidSet). On appelle *tidset* tout sous-ensemble d'identificateurs de transactions (tids) de $\mathcal{T}$. Pour simplifier, on écrira un tidset sans les accolades et sans les virgules séparant les éléments de l'ensemble.

Exemple. Le tidset $\{1, 3\}$, noté 13, représente l'ensemble des identificateurs des transactions 1 et 3 de la base $\mathcal{D}$ de l'exemple de la Figure 4.1.

Définition 6. (Treillis). Un ensemble ordonné $(T, \leq)$ est un treillis si toute paire d'éléments de T possède une borne inférieure et une borne supérieure. On désigne la borne inférieure de la paire (x, y) par $(x \wedge y)$ et la borne supérieure par $(x \vee y)$. On note la borne inférieure de T par $\perp$ et la borne supérieure par $\top$.

Définition 7. (Treillis $(\mathcal{P}(I), \subseteq)$). L'ensemble $\mathcal{P}(\mathcal{I})$ des parties d'un ensemble $\mathcal{I}$ muni de l'inclusion $\subseteq$ est un treillis. Les opérations binaires $\wedge$ et $\vee$ sont respectivement $\cap$ et $\cup$. De plus, ce treillis admet une borne inférieure $\perp = \emptyset$ et une borne supérieure $\top = \mathcal{I}$.

Définition 8. (Base de transactions). Une base de transactions $\mathcal{D}$ est un ensemble de couples formés d'un identificateur de transaction *tid* et de la transaction proprement dite. On l'appelle aussi *contexte d'extraction*.

$$\mathcal{D} = \{(y, X_y)/y \in \mathcal{T}, X_y \subseteq \mathcal{I}\}$$

Exemple. L'exemple de la Figure 4.1 représente une base de transactions avec :
- $\mathcal{I} = \{a, b, c, d\}$
- $\mathcal{T} = \{1, 2, 3, 4, 5\}$
- $\mathcal{D} = \{(1, ab), (2, ac), (3, cd), (4, bcd), (5, abcd)\}$

Une base de transactions peut être représentée sous forme horizontale (Figure 4.2 (a)), verticale (Figure 4.2 (b)) ou binaire (Figure 4.2 (c)).

1	a	b		
2	a	c		
3	c	d		
4	b	c	d	
5	a	b	c	d

(**a**)

a	b	c	d
1	1	2	3
2	4	3	4
5	5	4	5
		5	

(**b**)

	a	b	c	d
1	1	1	0	0
2	1	0	1	0
3	0	0	1	1
4	0	1	1	1
5	1	1	1	1

(**c**)

FIGURE 4.2 – Différentes représentations d'une base de transactions

Définition 9. (Applications t et i). On définit deux applications t et i comme suit : pour un motif X, t(X) représente l'ensemble tidset des transactions contenant X.
$t : \mathcal{P}(\mathcal{I}) \rightarrow \mathcal{P}(\mathcal{T})$
$X \mapsto t(X) = \{y \in \mathcal{T} | \exists X_y, (y, X_y) \in \mathcal{D} \wedge X \subseteq X_y\}$
Pour un tidset de Y, les items communs à toutes les transactions identifiées par les tids de Y sont donnés par l'application i, duale de t.
$i : \mathcal{P}(\mathcal{T}) \rightarrow \mathcal{P}(\mathcal{I})$
$Y \mapsto i(Y) = \{x \in \mathcal{I} | \forall (y, X_y) \in \mathcal{D}, y \in Y \Rightarrow x \in X_y\}$

Les applications t et i possèdent les propriétés suivantes :
si X, X_1, X_2 sont inclus dans $\mathcal{I}$ et Y, Y_1, Y_2 sont inclus dans $\mathcal{T}$, on a alors :
- $X_1 \subseteq X_2 \Rightarrow t(X_2) \subseteq t(X_2)$
- $Y_1 \subseteq Y_2 \Rightarrow i(Y_2) \subseteq i(Y_1)$
- $Y \subseteq t(X) \Rightarrow X \subseteq i(Y)$
- $t(X_1 \cup X_2) = t(X_1) \cap t(X_2)$

Exemple. $t(bd) = t(b \cap t(d)) = 145 \cap 345 = 45$
$i(45) = i(4) \cap i(5) = bcd \cap abcd = bcd$

Définition 10. (Fréquence). La fréquence d'un motif X, noté $freq(X)$, est le nombre de transactions de $\mathcal{D}$ contenant X tel que :

$$freq(X) = |\{(y, X_y) \in \mathcal{D} / X \subseteq X_y\}| = |t(X)|$$

Exemple. Dans l'exemple cité dans la Figure 4.1, on a freq(ab)= 2, vu que le motif ab apparaît dans les transactions 1 et 5 de $\mathcal{D}$.

Définition 11. (Support). Le support d'un motif X, noté $supp(X)$ est la proportion de transactions de $\mathcal{D}$ contenant X :

$$supp(X) = \frac{|\{(y, X_y) \in \mathcal{D} / X \subseteq X_y\}|}{|\mathcal{D}|} = \frac{|t(X)|}{|\mathcal{D}|}$$

Le support prend sa valeur dans l'intervalle $[0, 1]$ et il est souvent exprimé en pourcentage.

Exemple. Dans l'exemple de la Figure 4.1, on a supp(ab)= 0.4 (40%) vu que le motif ab apparaît dans deux transactions parmi 5 de $\mathcal{D}$.

Définition 12. (Motif fréquent). Étant donné un seuil σ, appelé support minimum, un motif X est dit fréquent (relativement à σ) dans une base de transactions $\mathcal{D}$, si son support dépasse un seuil fixé a priori appelé support minimum et noté σ.

$$\text{X est fréquent ssi } supp(X) \geq \sigma$$

Exemple. Dans l'exemple de la Figure 4.1, pour un support σ= 40%, le motif cd de support égal à $3/5 = 60\%$ est fréquent.

Motifs fréquents fermés. Soient deux applications t et i définies ci-dessus. L'application t associe à un motif X l'ensemble des transactions contenant X. L'application i associe à un tidset Y l'ensemble des items appartenant à toutes les transactions de Y. Une connexion de Galois est un couple d'applications (i, t) entre l'ensemble des *tidsets* et l'ensemble des motifs.
L'opérateur de fermeture de la connexion de Galois est la composition $i \circ t(X) = i(t(X))$ associant l'ensemble maximal d'items communs aux transactions contenant le motif X. On notera l'opérateur de fermeture de la connexion de Galois par $\boxdot$, on écrira alors : $\boxdot(X) = i \circ t(X) = i(t(X))$.

Cet opérateur possède les propriétés suivantes :
Soient $X, X_1, X_2 \subseteq \mathcal{I}$,

1. Extension : $X \subseteq \boxdot(X)$
2. Monotonicité : si $X_1 \subseteq X_2 \Rightarrow \boxdot(X_1) \subseteq \boxdot(X_2)$
3. Idempotence : $\boxdot(X) = \boxdot(\boxdot(X))$

Définition 13. Motif fermé. Un motif X est dit fermé s'il est égal à sa fermeture :

$$X = \boxdot(X)$$

La propriété importante sur laquelle reposent les algorithmes d'extraction des motifs fréquents fermés est que le support d'un motif quelconque est égal au support de sa fermeture :

$$supp(X) = supp(\boxdot(X))$$

Cela signifie que si l'on dispose des fermés, on dispose de tous les motifs ainsi que de leurs supports. On note l'ensemble des motifs fermés fréquents par $\mathcal{FF}$:

$$\mathcal{FF} = \{l \subseteq \mathcal{I} | \boxdot(l) = l \wedge supp(l) \geq \sigma\}$$

Considérons l'exemple de la Figure 4.1.

$\boxdot(a) = a$ fermé	$\boxdot(b) = b$ fermé
$\boxdot(c) = c$ fermé	$\boxdot(d) = cd$ non fermé
$\boxdot(ab) = ab$	$\boxdot(ac) = ac$ fermé
$\boxdot(bc) = bcd$ non fermé	$\boxdot(bd) = bcd$ non fermé
$\boxdot(cd) = cd$ fermé	$\boxdot(bcd) = bcd$ fermé

4.2.2 Algorithme Apriori

L'algorithme Apriori est le premier algorithme de recherche des motifs fréquents. L'algorithme *Apriori* fut proposé en 1994 [KMR+94, AMS+96]. Le but de l'algorithme est d'extraire tous les motifs fréquents à partir d'une base de transactions (contexte d'extraction). Ces motifs servent ensuite à dériver des règles d'associations. L'algorithme *Apriori* fonctionne en utilisant un principe très simple selon lequel un ensemble d'attributs binaires ne peut être plus fréquent qu'un de ses sous-ensembles. Le principe de l'algorithme *Apriori* est présenté dans Algorithme 6.

Algorithme 6 : Apriori($\mathcal{D}$, σ)

Entrées : Base de transactions $\mathcal{D}$, support minimum σ
Sortie : Ensemble des motifs fréquents $\mathcal{F}$
Début
  $\mathcal{F}_1 = \{1\text{-motifs fréquents}\}$
  Pour $(k = 2; \mathcal{F}_{k-1} \notin \emptyset; k++)$ **faire**
    $C_k =$ Apriori-Gen($\mathcal{F}_{k-1}$)
    Pour chaque *transaction* t *de* $\mathcal{D}$ **faire**
      $C_t =$ sous-ensembles$(C_k, t) | C_t = \{c \in C_k, c \subseteq t\}$
      Pour chaque *candidat* c **faire**
        $supp(c)++$
      Finpour
    Finpour
    $\mathcal{F}_k = \{C \in C_k | supp(c) \geq \sigma\}$
  Finpour
  Retourner $\mathcal{F} = \cup_k \mathcal{F}_k$
Fin

Algorithme 7 : Apriori-Gen($\mathcal{F}_{k-1}$)

Entrées : Ensembles des items fréquents de cardinal k
Sortie : C_k
Début
  $C = \{c = f_1 \cup f_2 \text{tels que } (f_1, f_2) \in \mathcal{F} \bowtie \mathcal{F}, card(c) = k+1\}$
  Pour chaque $c \subset C$ **faire**
    Pour chaque $s \subset c; card(s) = k$ **faire**
      Si $s \in \mathcal{F}$ **alors**
        $C_k = C \setminus \{c\}$
      Finsi
    Finpour
  Finpour
  Retourner C_k
Fin

L'algorithme reçoit en entrée une base de transaction $\mathcal{D}$ représentée sous forme booléenne (matrice binaire) et le support minimum σ . A chaque itération, il génère

un ensemble de conjonctions de propriétés binaires susceptibles d'être fréquentes, appelées candidats. Cette génération de candidats est effectuée en tenant compte des conjonctions effectivement fréquentes de l'itération précédente. L'algorithme *Apriori* détermine les fréquences de cet ensemble de candidats en lisant une par une les lignes du fichier de base. Les conjonctions suffisamment fréquentes sont retenues. Lorsqu'il n'y a plus de candidats générés, la boucle prend fin et les conjonctions retenues sont fournies comme résultat.

L'exemple ci-dessous montre le processus d'extraction des motifs fréquents sur la base de transactions $\mathcal{D}$ (Figure 4.1) pour un support $\sigma = 0.4$ correspondant à 2 transactions.

A la première itération de l'algorithme, chaque item de $\mathcal{I}$ est un 1-motif de C_1. Un premier parcours de $\mathcal{D}$ permet de trouver le support de chaque 1-motif. Tous les 1-motifs fréquents, c'est-à-dire les motifs ayant un support supérieur ou égal à 0.4 seront gardés dans $\mathcal{F}_1$. Afin de découvrir les 2-motifs fréquents, Apriori effectue dans la seconde itération une jointure de $\mathcal{F}_1$ avec $\mathcal{F}_1$ ($\mathcal{F} \bowtie \mathcal{F}$) pour trouver l'ensemble C_2 des candidats de taille 2. Seuls les 2-candidats n'ayant pas de sous-ensembles non fréquents sont gardés.

C_1
Motif
a
b
c
d
e

$\xrightarrow{\text{Parcours de } \mathcal{D}}$

C_1	
Motif	**Support**
a	3/5
b	3/5
c	3/5
d	4/5
e	3/5

$\rightarrow$

$\mathcal{F}_1$	
Motif	**Support**
a	3/5
b	3/5
c	3/5
d	4/5
e	3/5

Un second parcours de $\mathcal{D}$ est alors effectué pour déterminer le support de chacun des 2-motifs candidats, seuls les 2-motifs fréquents sont gardés dans $\mathcal{F}_2$. Ainsi le motif *ad* n'ayant pas de support suffisant est supprimé. Les 3-motifs sont obtenus en combinant les motifs de $\mathcal{F}_2$ deux à deux, c'est-à-dire par jointure $\mathcal{F}_2$ avec $\mathcal{F}_2$ ($\mathcal{F}_2 \bowtie \mathcal{F}_2$). Seuls les 2-motifs ayant le même préfixe de taille 1 sont générés. Par exemple les 2-motifs *ab* et *ac* forment le candidat *abc*. On s'assure également que les candidats générés n'ont pas de sous-ensembles non fréquents.

C_2
Motif
ab
ac
ad
bc
bd
cd

$\xrightarrow{\text{Parcours de } \mathcal{D}}$

C_2	
Motif	**Support**
ab	2/5
ac	2/5
ad	1/5
bc	2/5
bd	2/5
cd	3/5

$\rightarrow$

$\mathcal{F}_2$	
Motif	**Support**
ab	2/5
ac	2/5
ad	1/5
bc	2/5
bd	2/5
cd	3/5

Un troisième parcours de $\mathcal{D}$ est alors effectué pour déterminer les 3-motifs fréquents. De nouveau, on effectue la jointure $\mathcal{F}_3 \bowtie \mathcal{F}_3$ pour trouver l'ensemble C_4 des candidats de taille 4, qui est dans ce cas vide car on n'a plus qu'un seul élément

de taille 3. L'algorithme s'arrête alors après avoir trouvé tous les motifs fréquents et seulement les motifs fréquents.

C_3
Motif
abc
bcd

$\xrightarrow{\text{Parcours de } \mathcal{D}}$

C_3	
Motif	**Support**
abc	1/5
bcd	2/5

$\rightarrow$

$\mathcal{F}_3$	
Motif	**Support**
bcd	2/5

4.2.3 Algorithme Close

Close a été proposé par Pasquier et al. [PBTL99] pour l'extraction des motifs fréquents fermés, utilisant un mécanisme de fermeture basé sur l'analyse formelle des concepts (AFC). Il est basé sur la notion des motifs générateurs des motifs fermés. Les générateurs d'un motif fréquent fermé f sont les motifs minimaux g, au sens de l'inclusion, dont la fermeture est le motif f, c'est-à-dire $\boxdot(g) = f$. Ces motifs sont utilisés afin de construire un ensemble de motifs fermés candidats (motifs fermés potentiellement fréquents) qui sont les fermetures des générateurs.

Algorithme 8 : Close ($\mathcal{D}$, σ)

Entrées : Base de transactions $\mathcal{D}$, support minimum σ
Sortie : Ensemble des motifs fréquents fermés $\mathcal{FF}$
Début
  $\mathcal{FFC}_1$.générateurs={1-motif} ;
  // $\mathcal{FFC}_k$: ensemble des k-groupes candidats des k-générateurs
  Pour *($k = 1$; $\mathcal{FFC}_k$.générateurs $\neq \emptyset$; $k++$)* **faire**
    Gen-Closure($\mathcal{FFC}_k$, $\mathcal{D}$)
    Pour chaque *groupe candidat* $c \in \mathcal{FFC}_k$ **faire**
      Si *(c.support $\geq \sigma$)* **alors**
        $\mathcal{FF}_k = \mathcal{FF}_k \cup \{c\}$
        // ensemble de k-groupes fréquents des k-générateurs
      Finsi
    Finpour
    $\mathcal{FFC}_{k+1}$ = Gen-Generator($\mathcal{FF}_k$)
  Finpour
  Retourner $\mathcal{FF} = \cup_k \mathcal{FF}_k$
Fin

Pendant chaque itération k de Close, un ensemble de k-générateurs candidats $\mathcal{FFC}_k$ est considéré. Chaque élément de cet ensemble est constitué de trois éléments : le k générateur candidat, sa fermeture qui est un motif fermé candidat, et leur support. A la afin de l'itération k, l'algorithme *Close* stocke un ensemble $\mathcal{FF}_k$ contenant les k-générateurs fréquents, leur fermeture qui sont des motifs fermés fréquents et leur support.

L'algorithme *Close* (Algorithme 8) commence par initialiser l'ensemble $\mathcal{FFC}_1$ des 1-générateurs avec la liste des 1-motifs. Ensuite pour chaque itération k :

1. la fermeture de chaque k-générateur et le support du générateur et de sa fermeture sont calculés ;

2. pour chaque k-générateur fréquent, sa fermeture et son support sont insérés dans l'ensemble des motifs fermés fréquents.
3. un ensemble de (k+1)-générateurs est construit en utilisant les k-générateurs fréquents de l'ensemble des motifs fermés fréquents.

Algorithme 9 : Gen-Closure ($\mathcal{FFC}_k$, $\mathcal{D}$)

Entrées : Base de transactions $\mathcal{D}$
Ensemble des k-groupes candidats des k-générateurs $\mathcal{FFC}_k$
Sortie : $\mathcal{FFC}_k$ mis à jour
Début
  $\mathcal{FFC}_k$.fermé $= \emptyset$
  $\mathcal{FFC}_k.supports = 0$
  Pour chaque *transaction* $t \in D$ **faire**
    $G_0 = Subset(\mathcal{FFC}_k.\text{générateur}, i(\{t\}))$
    Pour chaque *générateur* $g.générateur \in G_0$ **faire**
      Si *(g.fermé= $\emptyset$)* **alors**
        alors $g.ferm = i(\{t\})$
      alors
        g.fermé = g.fermé $\cap\ i(\{t\})$
      Finsi
      $g.support++$
    Finpour
  Finpour
  Retourner $\cup\{g \in \mathcal{FFC}_k | g.fermé = \emptyset\}$
  `// Champs fermés et supports des candidats` $\mathcal{FFC}_k$ `mis à jour`
Fin

Algorithme 10 : Gen-Generator ($\mathcal{FF}_k$)

Entrées : Ensemble de k-groupes fréquents de k-générateurs $\mathcal{FF}_k$
Sortie : Champs générateurs initialisés $\mathcal{FFC}_{k+1}$
Début
  $\mathcal{FFC}_{k+1}$.générateurs $= Apriori(\mathcal{FF}_{k.\text{générateurs}})$
  Pour chaque *générateur* $g.générateur \in \mathcal{FFC}_{k+1}$ **faire**
    $S_g = (\mathcal{FF}_k.\text{générateurs}, g.\text{générateur})$
    Pour chaque $s \in S_g$ **faire**
      Si *(g.générateur* $\subseteq$ *s.fermé)* **alors**
      Finsi
      Supprimer g de $\mathcal{FFC}_{k+1}$
    Finpour
  Finpour
  Retourner $\mathcal{FFC}_{k+1}$
  `// Champs générateurs initialisés`
Fin

La procédure *Gen-Generator* génère d'abord les (k+1)-générateurs candidats en joignant les k-générateurs de $\mathcal{FF}_k$ possédant les mêmes (k-1) premiers items. Les (k+1)-générateurs candidats dont on sait qu'ils sont soit non fréquents, soit non

minimaux sont ensuite supprimés. Ce sont les générateurs ayant au moins un sous-ensemble n'appartenant pas à $\mathcal{FF}_k$. Enfin on supprime parmi les générateurs ceux dont la fermeture a déjà été calculée. Un tel générateur est identifié car il est inclus dans la fermeture d'un k-générateur fréquent de $\mathcal{FF}_k$ dont il est un sur-ensemble.

Nous présentons dans la Figure 4.3, un exemple d'exécution de l'algorithme Close toujours sur l'exemple de la base de transactions $\mathcal{D}$ (Figure 4.1).

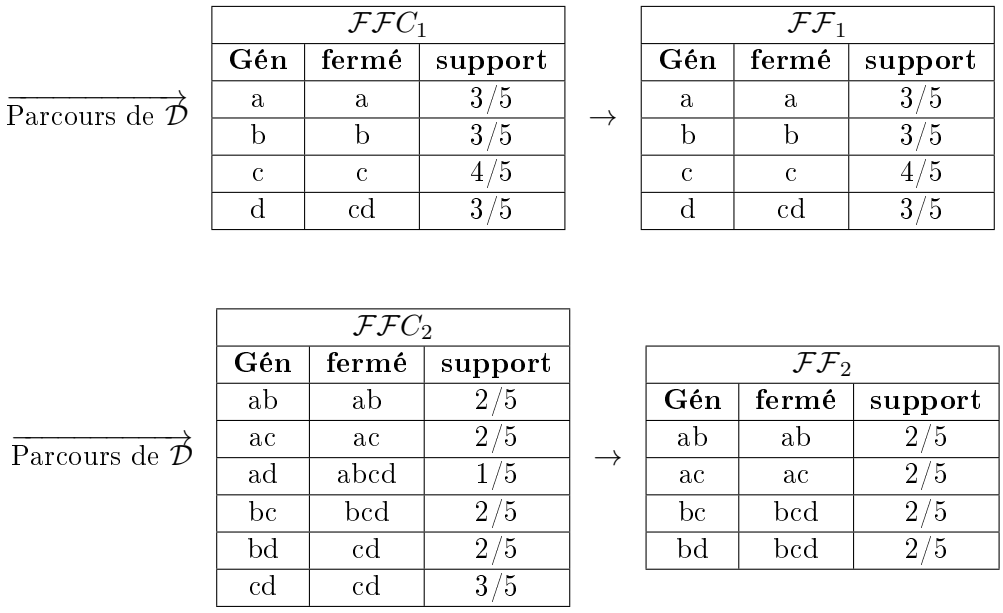


FIGURE 4.3 – Exemple d'exécution de *Close* sur la base de transactions $\mathcal{D}$

4.3 Requête décisionnelle

Les requêtes décisionnelles que nous considérons ici sont des requêtes SQL utilisant des opérateurs OLAP. Nous utilisons plus précisément le modèle de requêtes défini dans la Figure 4.4 [DBB05] comme étant une grammaire qui représente un sous-ensemble du standard SQL, introduisant plus de capacités analytiques à l'interrogation des bases de données relationnelles.

Définissons la sémantique de la terminologie employée dans la Figure 4.4.

- 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.

Query : :-	
Select	![<Attribute Clause>|<Aggregate Clause>
	|[<Attribute Clause>, <Aggregate Clause>]]
From	!<TableClause>[<Where Clause>
	|| [<Group by Clause>*<Having Clause>]]
Attribute Clause : :-	Descriptor-Attribute Name[[,<Attribute Clause>]|⊥]
Aggregate Clause : :-	![Aggregate Function Name(Measure-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**|**Grouping Sets**]<Attribute Clause>
Having Clause : :-	[Alias| Aggregate Function Name (Measure-Attribute Name)]
	<Comparison Operator>[Attribute Value | Attribute Value List]

Figure 4.4 – Grammaire de requêtes décisionnelles

4.4 Recommandation interactive de requêtes décisionnelles

4.4.1 Principe général

Pour améliorer le processus d'analyse dans les entrepôts de données, nous définissons une approche de recommandation de requêtes décisionnelles différente des approches existantes dans la littérature. En effet, notre approche a pour objectif d'aider l'utilisateur à formuler de nouvelles requêtes décisionnelles de manière interactive. Les requêtes décisionnelles auxquelles nous nous intéressons sont des requêtes décisionnelles de la forme "Select ... From ... Where ... Group By Cube (rollup)...". Ainsi, nous proposons une approche de recommandation qui aide l'utilisateur à construire sa requête, pas à pas, en lui suggérant des attributs fréquemment utilisés tout en respectant leur appartenance aux différentes *Clauses* (Select, Where, Group By, etc.) dans les requêtes précédentes.

Pour partager l'expérience des autres utilisateurs, notre approche exploite l'historique des requêtes de l'ensemble de tous les utilisateurs du système décisionnel.

Nous pensons que la pertinence d'une recommandation est fortement corrélée avec la fréquence de son utilisation dans l'ensemble des requêtes soumises par les différents utilisateurs. C'est pourquoi, nous avons recours aux techniques de fouille de données.

Notre approche de recommandation de requêtes se déroule en trois phases. (1) Tout d'abord, à partir d'un ensemble de requêtes issues des fichiers logs des utilisateurs, un prétraitement est nécessaire pour construire le contexte d'extraction des motifs fréquents (matrice "requêtes-attributs"). (2) Nous procédons ensuite à l'application de l'algorithme d'extraction des motifs fréquents sur la matrice "requêtes-attributs". (3) Nous exploitons les résultats obtenus (motifs fréquents) pour recommander à l'utilisateur des attributs fréquemment utilisés pour l'aider à anticiper dans l'écriture d'une nouvelle requête décisionnelle. Par conséquent, l'utilisateur peut construire progressivement sa requête en choisissant ses attributs parmi les suggestions faites par le système. La requête ainsi obtenue pourra être soumise par l'utilisateur à l'entrepôt de données pour construire un nouveau cube OLAP correspondant à un nouveau contexte d'analyse.

L'approche que nous proposons s'intègre parfaitement dans le processus habituel d'interrogation des entrepôts de données. Cette intégration est non intrusive et l'utilisateur peut décider ou de ne pas accepter les suggestions du système. Il s'agit d'une approche collaborative de recommandation de requêtes décisionnelles puisqu'elle permet à l'utilisateur de soumettre à l'entrepôt de données des requêtes pertinentes construites, pas à pas, à partir des attributs les plus fréquemment utilisés par l'ensemble des acteurs de la communauté d'utilisateurs à laquelle il appartient. Le processus général que nous proposons pour notre approche est illustré par la Figure 4.5.

4.4.2 Processus de recommandation interactive de requêtes décisionnelles

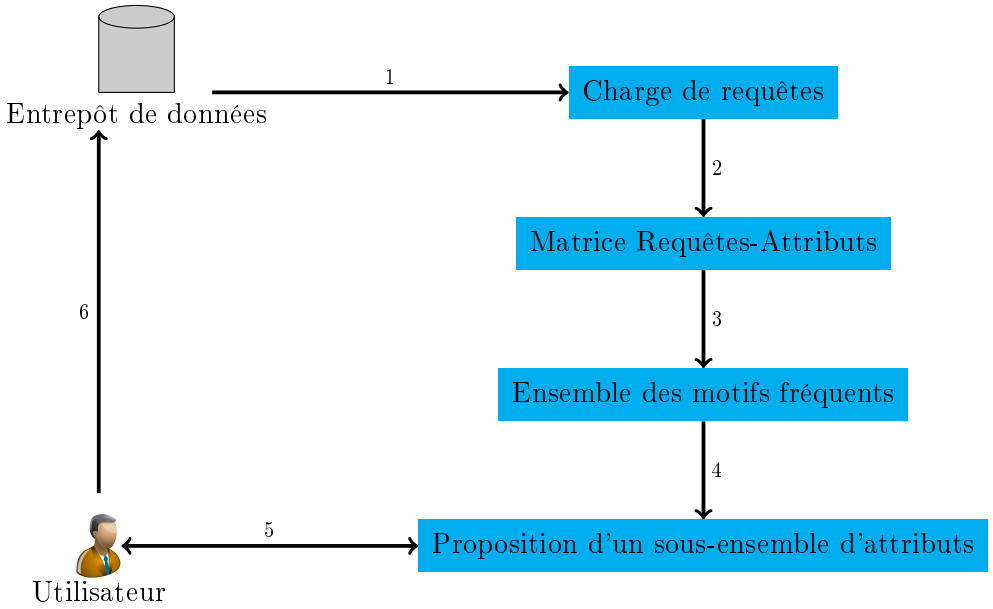
Nous présentons, dans cette section, les différentes étapes de notre approche d'aide à la construction de requêtes décisionnelles pertinentes.

4.4.2.1 Extraction de la charge de requêtes

La première étape de notre stratégie de recommandation de requêtes décisionnelles consiste en l'extraction d'une charge de requêtes des différents utilisateurs de l'entrepôt de données à partir des fichiers logs de requêtes. En effet, nous extrayons du journal des transactions les requêtes adressées au SGBD dans lequel est stocké l'entrepôt de données.

4.4.2.2 Extraction des items à partir d'une charge de requête

La deuxième étape de notre stratégie de recommandation consiste à extraire tous les items utilisés dans les requêtes en respectant la clause à laquelle ils appartiennent. En effet, les requêtes SQL présentes dans la charge de requêtes sont traitées par un analyseur syntaxique afin d'en extraire tous les attributs susceptibles d'être recommandés. Ces attributs sont extraits des différentes requêtes de l'utilisateur à partir des clauses SELECT, FROM, WHERE, GROUP BY et HAVING. Les différents



1. Extraction des requêtes à partir du fichier log
2. Application de l'algorithme Close
3. Extraction des attributs fréquents
4. Génération de recommandations d'attributs à partir des motifs fréquents
5. Dialogue entre l'utilisateur et le système de recommandation
6. Soumission de la requête

Figure 4.5 – Processus général de recommandation de requêtes décisionnelles

items sont présentés dans le Tableau 4.1 et sont de la forme suivante : <nom de la clause>.<item> avec nom de la clause = {Select, From, Where, Group by, Having } et item= {attribut, mesure, dimension, prédicat}. Dans ce chapitre, nous nous intéressons aux attributs et nous tenons compte de la différenciation des attributs : attributs descriptifs (dimensions) et attributs mesures (table de faits). Dans la grammaire de requêtes décisionnelles présentée dans la Figure 4.4, on désigne attribut descriptif par Descriptor-Attribute Name et l'attribut mesure par Measure-Attribute Name. On les note respectivement par D-Attribut et M-Attribut.

Item	**commence par**	**se termine par**
D-Attribut/M-Attribut	SELECT	FROM
Dimension/fait	FROM	WHERE, GROUP BY, fin de la requête (;)
D-Attribut Where	WHERE	GROUP BY, fin de la requête (;)
D-Attribut Group By	GROUP BY	HAVING, fin de la requête (;)
M-Attribut Having	HAVING	fin de la requête (;)

Tableau 4.1 – Identification des différents items

Afin d'aider l'utilisateur à formuler sa requête décisionnelle, dans un premier temps, notre idée était de suggérer tous les items qu'on peut extraire de la charge de requêtes (attribut, mesure, prédicat) [KBB12, KB12, KB13]. Nous avons constaté l'importance des attributs descriptifs et mesures. Ainsi, nous recommandons que les

D-attributs en fonction de M-attributs choisis.

4.4.2.3 Construction du contexte d'extraction

Une caractéristique majeure de requête décisionnelle est son agrégation de mesures par une ou plusieurs dimensions comme l'une des principales opérations, par exemple, le calcul et le classement des ventes totales par pays (ou par année). Autres opérations populaires comprennent la comparaison de deux mesures (par exemple, les ventes et le budget) agrégés par les mêmes dimensions. Ainsi, les requêtes elles-mêmes renvoient des résultats calculés sur les attributs de mesure. Chacune des mesures numériques dépend d'un ensemble de dimensions, qui fournissent le contexte de la mesure. Les dimensions sont supposées déterminer de manière unique la mesure [CD97]. Par conséquent, les mesures ont une importance extrême dans notre approche de recommandation de requêtes. De ce fait, nous proposons une recommandation des attributs descriptifs liée à la mesure.

Nous regroupons les requêtes utilisant la/les même(s) mesure(s). Nous représentons les différentes combinaisons possibles de ces mesures sous forme de treillis comme illustré dans la Figure 4.6. Chaque nœud dans le treillis de mesures représente une combinaison de mesures.

Le nombre de niveaux dans un treillis est égal au nombre total de mesures utilisées. La taille du treillis croît de façon exponentielle avec le nombre d'attributs total. Toutefois, généralement dans un entrepôt de données, le nombre de mesures est très faible. Soit N le nombre de nœuds dans un treillis, alors $N = (2^A - 1)$ où A représente le nombre de mesures.

Pour chaque nœud du treillis représentant une mesure (M-attribut) ou une combinaison de mesures, nous construisons le contexte d'extraction qui est présenté par une matrice "requêtes-D-attributs" qui a pour lignes les requêtes de la charge utilisant cette mesure ou cette combinaison de mesures et pour colonnes les D-attributs précédés par le nom de la clause à laquelle ils appartiennent. L'existence d'un D-attribut dans une requête est symbolisée par un et son absence par zéro. Nous illustrons la construction de cette matrice à travers un exemple dans la section 4.5.

4.4.2.4 Recherche des motifs fréquents

La recherche des motifs fréquents est réalisée hors ligne sur la matrice "requêtes-D-attributs". Dans un premier temps, nous avons utilisé l'algorithme *Apriori* [KB12]. Nous avons constaté que l'algorithme *Apriori* générait plusieurs candidats. Par conséquent, plus le nombre de motifs fréquents est élevé, plus on génère de recommandations candidates. Pour éviter la prolifération des recommandations candidates, nous utilisons l'algorithme *Close*. Ainsi, l'application de l'algorithme *Close* sur le contexte d'extraction génère l'ensemble des motifs fréquents (et leurs supports) pour un support minimal fixé par l'utilisateur. Nous considérons cet ensemble comme notre configuration de recommandations candidates.

Dans le cadre de nos travaux, les éléments sont des requêtes et les items sont les D-attributs extraits de ces requêtes. La charge de requêtes peut être volumineuse. L'algorithme *Close* s'avère adapté à cette volumétrie. En effet, *Close* est basé sur les opérateurs de fermeture de Galois, qui permettent la détermination efficace des

ensembles fermés, selon cette fermeture, dans les grandes bases de données. En comparaison des autres algorithmes existants de détermination des ensembles fermés, *Close* permet de réduire le nombre d'accès aux données du contexte d'extraction, qui constituent les opérations les plus coûteuses en temps lorsque ces algorithmes sont appliqués à de grands volumes de données [PBTL99].

Par ailleurs, l'utilisateur d'un entrepôt de données suit en général un raisonnement logique dans un processus d'interrogation et d'analyse. Les données interrogées dans une session d'analyse sont souvent corrélées. De ce fait, les requêtes de la charge, prises dans un intervalle continu de temps, le sont aussi. Cette corrélation entre les requêtes donne lieu à un contexte d'extraction dense. Dans ce cas, l'algorithme *Close* est performant en temps de calcul et en espace mémoire nécessaires à la recherche des motifs fréquents. De plus, l'ensemble des motifs fermés fréquents étant un sous-ensemble de l'ensemble des motifs fréquents, le nombre d'opérations nécessaires à leur extraction est inférieur au nombre d'opérations nécessaires à la recherche des motifs fréquents. D'une part, cela a pour effet de réduire considérablement le temps de calcul des motifs fréquents fermés par rapport au temps de calcul des motifs fréquents. D'autre part, le nombre de recommandations candidates dépend du nombre de motifs fréquents. Plus le nombre de motifs fréquents est élevé, plus on génère de recommandations candidates. Ainsi, pour éviter la prolifération des recommandations candidates, il est judicieux de générer ces recommandations à partir des motifs fréquents fermés. Cela permet de réduire la complexité du problème de recommandation d'attributs. En appliquant l'algorithme *Close* (Algorithme 8) avec un support minimum = 60%, nous obtenons que les fréquents fermés $\mathcal{FF}$. Nous rappelons qu'un motif X est dit fréquent ssi $supp(X) \geq \sigma$. Dans ce cas tous motifs ayant un support $\geq 60\%$ sont fréquents. Toutefois, l'algorithme *Close* ne se base pas juste sur la fréquence des motifs mais aussi leurs fermeture. Il cherche les motifs fréquents fermés. Un motif est dit fermé s'il est égal à sa fermeture $X = \boxdot(X)$. En effet, la propriété importante sur laquelle repose l'algorithme *Close* est que le support d'un motif quelconque est égal au support de sa fermeture : $supp(X) = supp(\boxdot(X))$. Cela signifie que si l'on dispose des fermés, on dispose de tous les motifs ainsi que de leurs supports. Par conséquent, l'output de cette étape est les motifs fréquents fermés $\mathcal{FF}$ qui seront exploités durant la recommandation de requêtes décisionnelles.

4.4.2.5 Génération des recommandations

Nous présentons ici la mise en correspondance entre les motifs fréquents fermés et les D-attributs candidats à la recommandation. Nous nous focalisons sur la façon exacte dont les recommandations sont calculées dans notre système de recommandation exploitant les motifs fréquents.

Généralement, les SID doivent fournir de façon synthétique et simple les éléments nécessaires aux décideurs pour évaluer une situation. Dans ce contexte, il nous semble indispensable de mettre en relief l'aspect interactif et de viser à produire des résultats adaptées aux besoins spécifiques de chaque utilisateur. En effet, notre système accompagne le décideur durant la formulation de sa requête décisionnelle. Il a pour objectif de fournir des résultats répondant aux besoins et attentes des utilisateurs et ne nécessitant aucun effort et traitement supplémentaires pour être examinés et

exploités. Ainsi, notre système de recommandation est interactif puisqu'il offre la possibilité d'intervention à l'utilisateur tout au long de la formulation de sa requête. En effet la notion d'interactivité renvoie au rôle indispensable de l'utilisateur dans son fonctionnement, rôle non passif qui sous-tend le terme "Aide à la Décision", mais aussi à la qualité de la collaboration du système avec le décideur. Tout d'abord, nous proposons à l'utilisateur de choisir le/les M-attribut(s) à utiliser. Supposons que l'utilisateur choisit $M1$, le système va utiliser alors le contexte d'extraction qui correspond à cette mesure c'est-dire la matrice binaire du nœud $M1$ du treillis de mesures. Dès que l'utilisateur commence à formuler sa requête par la saisie du premier attribut A dans la clause Select, le système cherche l'attribut A dans les différents motifs fréquents fermés $\mathcal{FF}$ obtenus dans le nœud $M1$ du treillis de mesures. Plus précisément, le système va chercher les D-attributs précédés par *Select* c'est-à-dire les D-attributs qui appartiennent à la clause Select. En effet, chaque D-attribut proposé est associé à la clause dans laquelle il apparaît. Il propose alors à l'utilisateur la liste de D-attributs appartenant aux motifs contenant A dans la clause Select. L'utilisateur choisit alors un attribut A_1 parmi les attributs les plus fréquemment associés à A (à partir des motifs $\mathcal{FF}$) ou introduit de nouveaux attributs. Ainsi, avec chaque saisie, une nouvelle représentation est calculée et affichée ensuite à l'utilisateur. Enfin, lorsque l'utilisateur termine l'exploration des recommandations, l'entrepôt de données est interrogé avec les attributs choisis par l'utilisateur.

4.5 Exemple illustratif

Pour illustrer nos propos, nous prenons le cas de l'entrepôt de données *Foodmart* dont un extrait de son schéma logique en étoile est donné dans la Figure 3.1. En se basant sur cet entrepôt de données et une charge de requêtes, nous montrons dans cette section comment fonctionne notre système de recommandation.

Tout d'abord, nous nous intéressons au fait *sales_fact_1998* qui comporte trois mesures *Store_sales*, *Store_cost* et *Unit_sales*. Ces mesures peuvent être étudiées selon cinq dimensions *Product*, *Store*, *Customer*, *Time_by_day* et *Promotion* comme illustré dans la Figure 3.1. Il s'agit d'observer les ventes par rapport aux clients, promotion, magasin, produit et temps. L'information sur les produits est collectée dans la dimension *Product*, tandis que les informations reliées au temps sont sauvegardées dans la dimension time_by_day. Les informations concernant les clients (noms et adresses) sont sauvegardés dans la dimension *Customer*. Les attributs *Education*, *Gender*, *Marital_Status*, *Occupation* et *Yearly_Income* fournissent des informations additionnelles sur les clients. De plus, l'information sur les magasins d'alimentation individuels est collectée dans la dimension *Store*. Elle inclut les attributs *store_location*, *name*, *manager*, *size*, et *store_type*. Enfin, les informations concernant les promotions sont sauvegardées dans la dimension *Promotion*.

4.5.1 Prétraitement

Le pré-traitement du log de requêtes consiste à une tâche réalisée hors ligne contenant trois phases.

4.5.1.1 Phase 1 : Construction du treillis de mesures

Dans l'entrepôt de données "Foodmart", nous avons 7 mesures : *Store_ sales*, *Store_ cost*, *Unit_ sales*, *Amount*, *Warehouse_ sales*, *Warehouse_ cost* et *Store_ invoice*. Dans cet exemple, nous nous sommes intéressés au cube de données *Sales* (Figure 3.1). Ainsi, nous utilisons seulement les mesures *Store_ sales*, *Store_ cost* et *Unit_ sales*. Nous représentons les différentes combinaisons possibles de ces trois mesures dans le treillis dans la Figure 4.6.

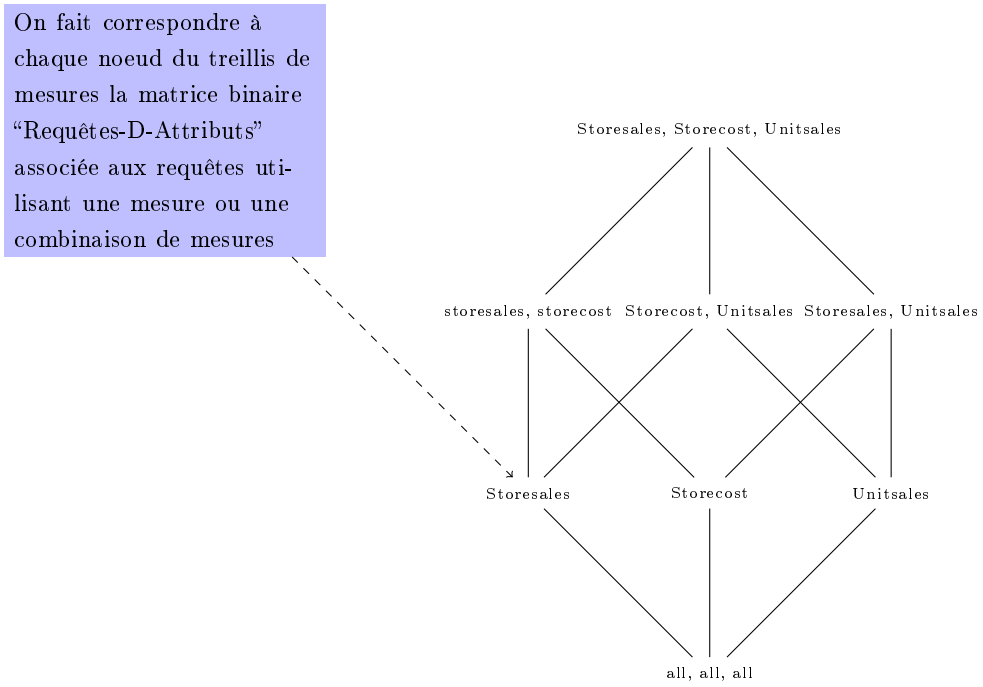


Figure 4.6 – Treillis de mesures

4.5.1.2 Phase 2 : Construction du contexte d'extraction

Pour chaque nœud du treillis représentant une mesure ou une combinaison de mesures, nous construisons la matrice binaire "requêtes-D-attributs" à partir de la charge de requête. Tout d'abord, nous procédons à l'extraction des attributs de chaque requête. Supposons que nous avons une charge contenant 3 requêtes q_1, q_2 et q_3 ayant comme attribut mesure Store_Sales. Notre matrice "requêtes-D-attributs" correspond au nœud du treillis Store_Sales. Ces trois requêtes sont associées respectivement à un ensemble d'attributs A_{q_1}, A_{q_2} et A_{q_3}. Pour construire la matrice "requêtes-D-attributs", les attributs sont précédés par le nom de la clause à laquelle ils appartiennent. Pour une meilleure clarté, l'ensemble de ces attributs est présenté dans un tableau (Tableau 4.2).

q_1 :
```
Select thedate, Product_ name, Store_ name, sum(Store_ sales)
From Sales, Time_by _day , Store, Product
Where Time_by _day .Time_id = Time_by _day .Time_id
And Sales.Store_id = Store.Store_id
And Sales.Product_id = Product.Product_id
And Store_Country = 'Canada'
And Product.brand = 'Jeffers'
Group by Grouping Sets (thedate, Store_name , Product_name) ;
```

q_2 :
```
Select Product_name , Promotion_district_id, Store_ name, thedate , avg(Store_Sales)
From Sales, Product, Promotion, Time_by _day, Store
Where Sales.Product_id = Product.Product_id
and Sales.Promotion_id = Promotion.Promotion_id
and Sales.Time_by_day.Time_id = Time_by_day.Time_id
and Promotion.media_type = 'TV'
and Store.store_Country = 'Mexico'
Group by Rollup (Product_name , Promotion_district_id, Store_ name, thedate) ;
```

q_3 :
```
Select Customer_id , Product_name, Store_city , sum(Store_Sales)
From Sales, Customer, Product, Store
Where Sales.Customer_id = Customer.Customer_id
and Sales.Product_id = Product.product_id
and Sales.Store_id = Store.Store_id
and Customer.Customer_marital_status ='S'
and Store.Store_type = 'Supermarket'
and Store.Store_state = 'Yucatan'
Group by Cube (Customer_id , Product_name , Store_city) ;
```

Pour construire la matrice "requêtes-D-attributs", les attributs sont précédés par le nom de la clause à laquelle ils appartiennent. Pour une meilleure clarté, l'ensemble de ces attributs est présenté dans un tableau (Tableau 4.2).

À partir de A_Q, nous construisons une matrice "requêtes-D-attributs" qui a pour lignes les 3 requêtes de notre charge et pour colonnes les attributs. Concrètement, la valeur 1 dans une case indique que l'utilisateur a utilisé l'attribut dans sa requête et 0 dans le cas contraire. Dans notre exemple, la matrice "requêtes-D-attributs" obtenue après l'analyse de la charge est composée de 22 colonnes et de 3 lignes (Tableau 5.2). Pour des raisons de clarté et de lisibilité, nous avons désigné les 22 D-attributs respectivement par les lettres de A à V (Tableau 5.2).

4.5.1.3 Phase 3 : Application de l'algorithme Close

L'algorithme *Close* est appliqué pour chaque nœud du treillis. L'input de l'algorithme *Close* consiste en une matrice binaire générée dans la phase 2 et l'output consiste en un ensemble de motifs fréquents fermés.

L'application de l'algorithme Close donne lieu au $\mathcal{FF}_k$ qui représente comme expliqué dans la Section 4.2.3 l'ensemble de k-groupes fréquents des k-générateurs. Chaque élément possède 3 champs : générateur, fermé et support.

	Select	Where	Group by
A_{q_1}	Select.thedate Select.Product_name Select.Store_name	Where.Time_id Where.Store_id Where.Product_id Where.Store_Country Where.Product_brand	Group by.Store_name Group by.thedate Group by.Product_name
A_{q_2}	Select.Promotion_district_id Select.Product_name Select.Store_name Select.thedate	Where.Product_id Where.Promotion_id Where.Time_id Where.Promotion.media_type Where.Store_Country	Group by.Promotion_id Group by.Product_name Group by.Store_name
A_{q_3}	Select.Customer_id Select.Product_name Select.Store_city	Where.Customer_id Where.Product_id Where.Store_id Where.Customer_marital_status Where.Store_type Where.Store_state	Group by.Product_name Group by.Customer_id Group by.Store_city
A_Q	Select.thedate Select.Product_name Select.Store_name Select.Promotion_district_id Select.Customer_id Select.Store_city	Where.Product_id Where.Store_id Where.Store_Country Where.Product_brand Where.Promotion_id Where.Promotion.media_type Where.Time_id Where.Customer_id Where.Customer_marital_status Where.Store_type Where.Store_state	Group by.thedate Group by.Store_name Group by.Product_name Group by.Promotion_id Group by.Store_city

Tableau 4.2 – D-attributs

4.5.2 Aide à l'écriture interactive de requêtes

Le principe de base qui sous-tend notre système de recommandation est les complétions de la requête courante. En effet, après la sélection de mesures par l'utilisateur, notre système lui propose des attributs de dimension. Supposons qu'un utilisateur sélectionne la mesure *Store_sales*.

Notre système utilisera alors les motifs fréquents fermés correspondant au nœud choisi dans le treillis. Supposons maintenant qu'il saisisse l'attribut *brand_name*, notre système va rechercher les motifs fréquents fermés correspondant aux mesures sélectionnées contenant cet attribut. Notre système proposera ainsi à l'utilisateur les autres attributs corrélés comme des complétions possibles pour sa requête. Si l'utilisateur accepte la suggestion d'un attribut, notre système va rechercher à nouveau les motifs fréquents fermés contenant cet attribut et ainsi de suite.

Requêtes \ D-attributs	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V
q_1	0	1	0	0	0	1	0	1	1	0	0	1	0	1	1	0	0	0	1	0	0	0
q_2	1	1	0	0	1	1	1	1	0	1	0	1	1	1	0	1	1	0	0	1	0	0
q_3	1	0	1	0	1	1	1	0	0	1	0	0	0	0	0	1	1	1	0	0	1	1

Tableau 4.3 – *Matrice "requêtes-D-attributs"*

Générateur	Fermé	Support
BC	ABC	2/3
GH	GHIJ	2/3
IJ	GHIJ	2/3

Tableau 4.4 – Ensemble des motifs fréquents fermés $\mathcal{FF}$

4.6 Développement et validation

4.6.1 Environnement de développement

Pour valider notre approche de recommandation, nous avons tout d'abord développé un prototype en Java, baptisé *FIMIOQR* (Frequent Itemset Mining for Interactive OLAP Query recommendation).

Notre système représente une couche logicielle intermédiaire (middleware-layer) au sommet du SGBD (SQL Server). Le fonctionnement de FIMIOQR est illustré dans la Figure 6.10. Les requêtes de l'utilisateur sont transférées via l'interface de l'entrepôt de données (Interface d'interrogation) à l'entrepôt de données et aussi à notre système. L'entrepôt de données traite la requête et retourne les résultats. En même temps la requête est enregistrée dans le log de requêtes. Ce log de requêtes est traité hors ligne pour en extraire les différents attributs de différentes requêtes. A chaque fois, l'utilisateur accède au système, le système de recommandation combine son input avec les motifs fréquents et génère un ensemble de recommandations qu'il lui suggère.

Notre système de recommandation est bien adapté à l'écriture pas à pas de la requête comme illustré dans les deux Figures 6.11 et 6.12. La Figure 6.11 montre les recommandations au niveau de la clause Select. La Figure 6.12 montre les recommandations au niveau de la clause Where. On trouvera plus de détails concernant le développement de l'outil FIMIOQR dans le Chapitre 6.

4.6.2 Expérimentation et validation

Pour valider notre approche d'aide à l'écriture de requêtes décisionnelles, nous avons mené des expériences et effectué des tests pour mesurer la qualité de nos recommandations et nous sommes par ailleurs intéressés à tester notre système. Ainsi, les tests que nous avons effectués sur *FIMIOQR* ont démontré que notre algorithme de recommandation donne de résultats satisfaisants. En effet, le temps de génération d'une recommandation ne dépasse pas la seconde. Nous avons observé que le temps d'exécution augmente quand la taille du fichier log augmente. Nous avons également observé que les recommandations générées par *FIMIOQR* sont de bonne qualité.

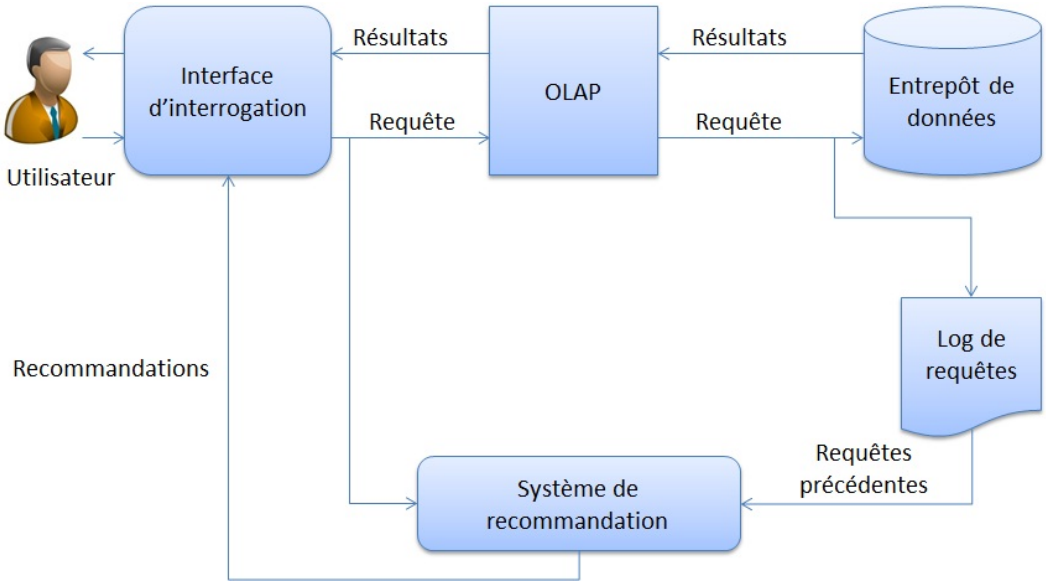


Figure 4.7 – Architecture de FIMIOQR

Tout d'abord, la première expérience évalue la performance de l'approche proposée pour suggérer des recommandations. Notre analyse de performance consiste à évaluer le temps nécessaire pour générer une recommandation. Le temps est calculé pour différentes tailles de log. La taille de log consiste au nombre total contenues dans ce log. Ainsi, la recommandation est calculée pour différentes tailles des logs. En effet, nous avons calculé le temps nécessaire de génération d'une recommandation pour différentes tailles de logs au niveau de l'assistance de l'écriture réalisée par le système FIMIOQR.

La Figure 4.10 montre que le temps nécessaire pour générer une recommandation est influencé par la taille des logs des requêtes. Il augmente linéairement avec la taille des logs mais reste toujours acceptable puisqu'il ne dépasse pas la seconde pour *FIMIOQR* (au niveau de l'écriture de la requête). Nous avons testé notre prototype sur une charge de 100 requêtes relatives à l'entrepôt de données *Foodmart* utilisé dans notre exemple. Nous avons enrichi, par ailleurs, la charge de requêtes de départ par les requêtes obtenues par FIMIOQR et qui sont recommandées à l'utilisateur. Comme nous pouvons le voir sur la Figure 4.10, il est évident que la tendance du temps d'exécution est à la hausse avec la taille du log de requêtes. Le temps d'exécution est acceptable avec la taille de 50 requêtes dans le log de requêtes.
La deuxième expérience que nous avons menée consiste à mesurer la qualité de la recommandation. Or, deux métriques, précision et rappel, sont souvent utilisées pour évaluer la qualité d'une recommandation [HK00]. En effet, un système de recommandation peut recommander des éléments intéressants ou non intéressants. La mesure de rappel indique l'efficacité d'une méthode pour localiser des éléments intéressants, tandis que la mesure de précision représente à quel point les instances recommandées par une méthode sont réellement intéressantes pour les utilisateurs. En bref, le rappel montre si les éléments intéressants peuvent être localisés c'est-à-dire cette

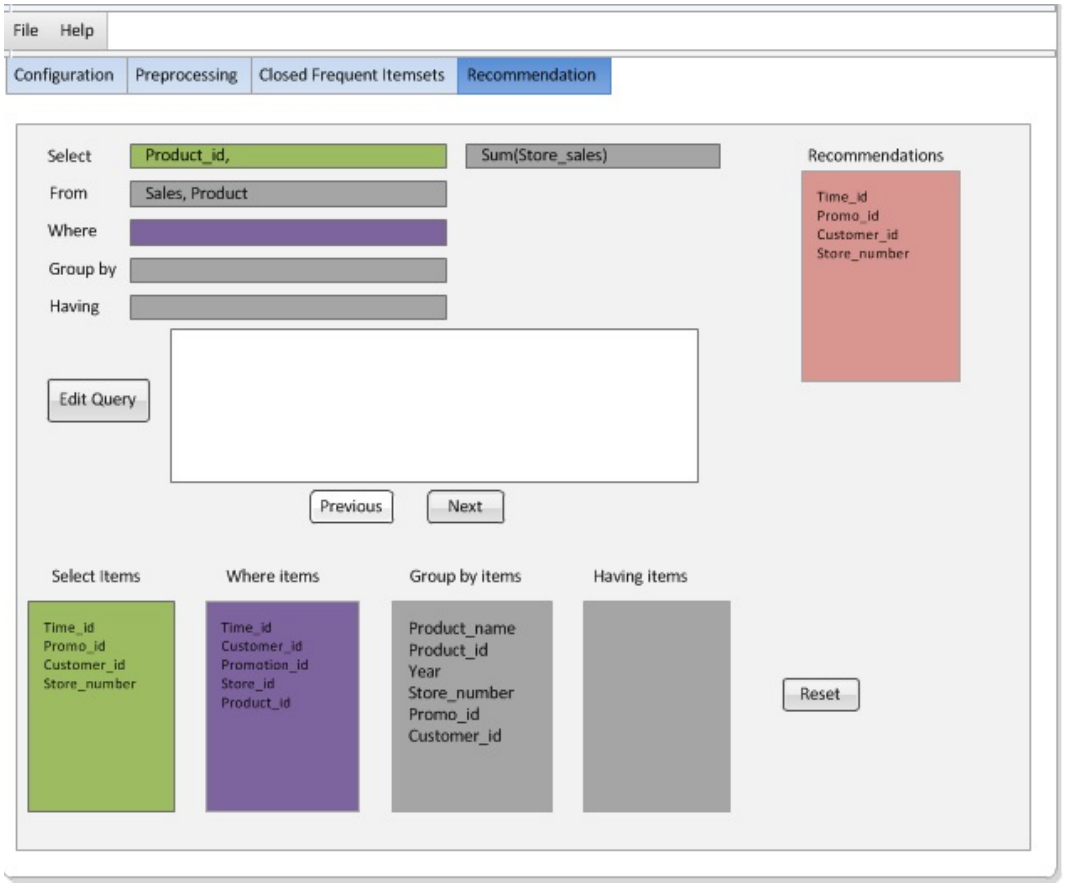


FIGURE 4.8 – FIMIOQR : clause Select

métrique calcule la probabilité qu'un élément pertinent soit choisi, et la précision indique si les éléments recommandés sont vraiment intéressants c'est-à-dire cette mesure représente la probabilité qu'un élément choisi soit pertinent.

Les éléments correctement recommandés = recommandations pertinentes ∩ recommandations émises.

La précision mesure la pertinence des recommandations émises. Une recommandation pertinente étant une recommandation faite sur un attribut déjà accepté par l'utilisateur dans l'ensemble de test. La formule est la suivante :

$$Précision = \frac{des\ éléments\ correctement\ recommandés}{nombre\ des\ éléments\ recommandés}$$

Le rappel mesure de son côté la capacité du système à faire des recommandations pertinentes à l'aide de la formule suivante :

$$Rappel = \frac{nombre\ des\ éléments\ correctement\ recommandés}{nombre\ éléments\ intéressants}$$

Cependant, dans notre cas, le nombre des éléments intéressants est fixé et égal au nombre des attributs utilisés dans l'entrepôt de données. La portée ou l'étendue des

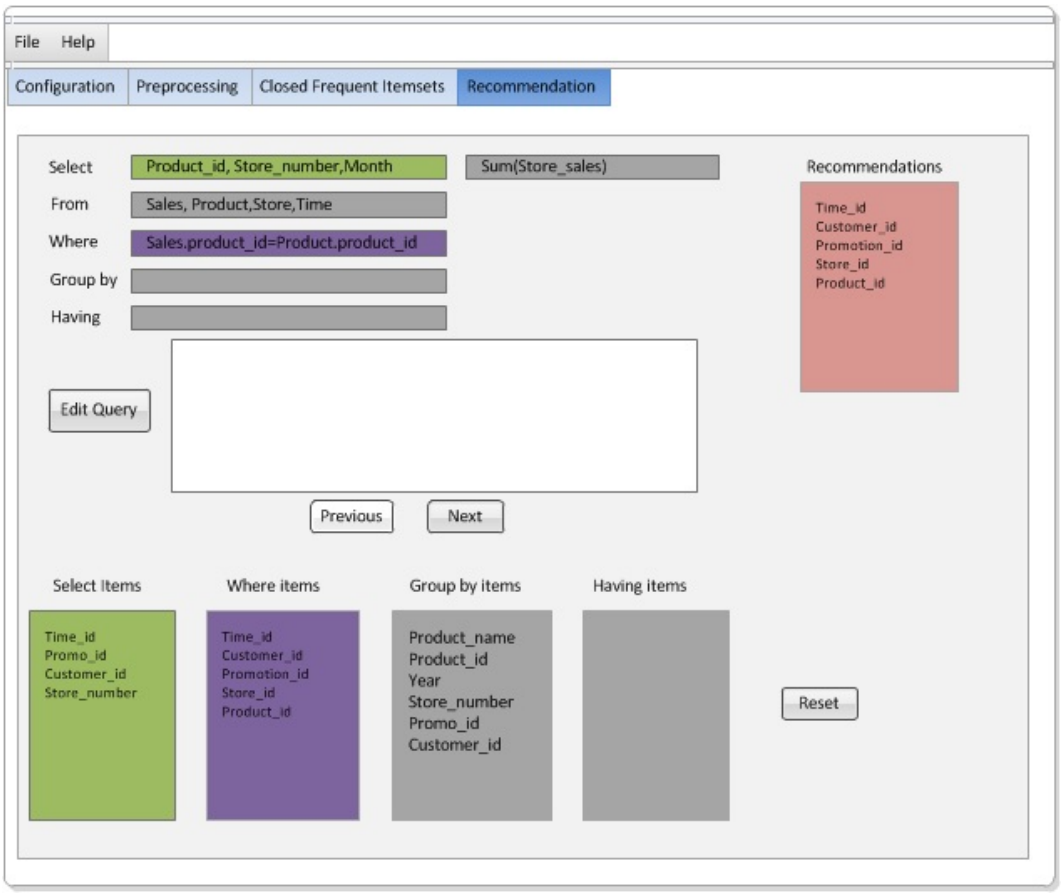


Figure 4.9 – FIMIOQR : clause Where

recommandations est limitée ; par conséquent, la métrique de rappel est non applicable dans notre cas. Nous pouvons utiliser seulement la métrique de précision pour évaluer la qualité de recommandations proposées. L'exactitude d'une recommandation peut être évaluée par la métrique de précision. La précision de chaque cas est calculé comme suit :

$$RP = \frac{nombre\ des\ attributs\ recommandés\ acceptés}{nombre\ des\ attributs\ recommandés}$$

Par ailleurs, l'évaluation de la performance du système est généralement basée sur une charge de requêtes. Lorsqu'une nouvelle requête est soumise, la charge doit être mise à jour. Dans ce contexte, nous proposons une mise à jour incrémentale de la charge.

La Figure 4.11 montre que la précision augmente légèrement avec la taille du log de requêtes. Cette figure démontre que les la majorité des requêtes courantes peuvent obtenir des recommandations réussies dans notre approche avec une précision comprise entre 0.14 et 0.58.

En outre, notre cadre de recommandation a été exécuté pour différentes valeurs du paramètre support minimal (minsup) de l'algorithme *Close*. Dans la pratique,

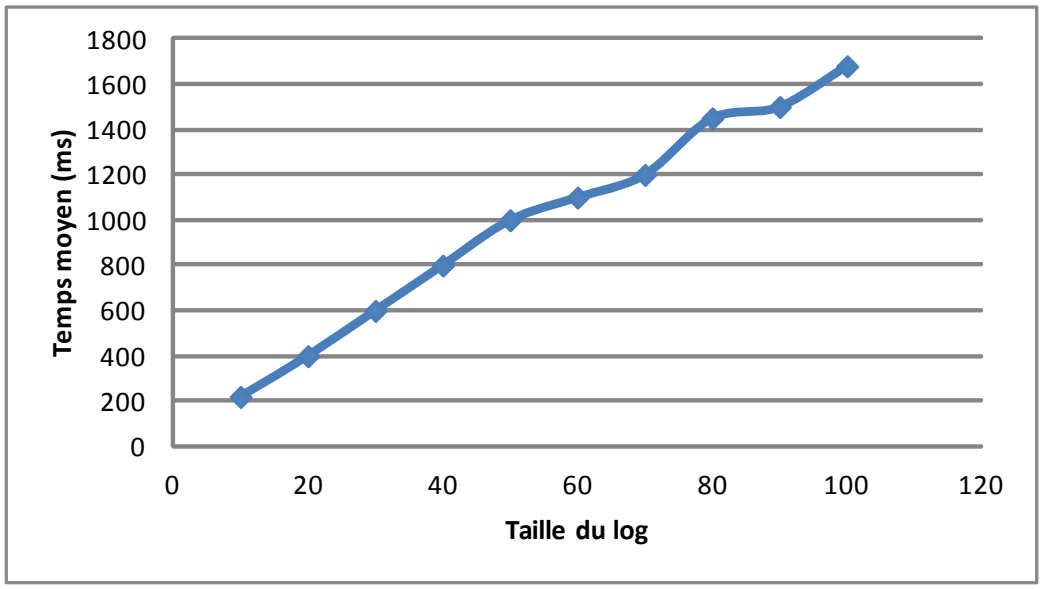


FIGURE 4.10 – Analyse de performance : Temps d'exécution en fonction de la taille du fichier log

ce paramètre nous permet de limiter le nombre de recommandations candidates à générer en sélectionnant uniquement celles qui sont les plus fréquemment utilisées par la charge de requêtes. La Figure 4.12 montre la précision de recommandation en fonction de la valeur du support minimal. Comme nous pouvons le voir dans la Figure 4.12, il est évident que la tendance de la précision de recommandation est à la hausse avec le paramètre minsup jusqu'à 60% où elle diminue.

4.7 Discussion

Dans le cadre de la personnalisation des entrepôts de données, notre approche s'inscrit dans le courant de la personnalisation par recommandations. Autrement dit la prise en compte de l'utilisateur est effectuée par la proposition des recommandations en se basant sur les logs des requêtes. L'idée d'employer les usages de l'utilisateur pour produire des recommandations est très populaire dans le domaine de la recherche d'information [AT05], et dans l'exploitation des usages du web (Web Usage Mining) [SCDT00]. Pour cela, notre contribution est d'adapter ces techniques existantes au domaine des entrepôts de données. Néanmoins, notre principale occupation est d'assister à l'écriture de la requête clause par clause et attribut par attribut. Cette assistance réside dans la génération des recommandations candidates en commençant par trouver quels motifs fréquents coïncident avec l'attribut courant et les présenter à l'utilisateur. L'avantage de notre approche est l'utilisation de la fouille de données en appliquant la méthode de recherche des motifs fréquents.

Nos travaux se rapprochent des travaux qui proposent des approches d'affinement de la requête utilisateur dans le domaine des bases de données [KKBS10] présenté en

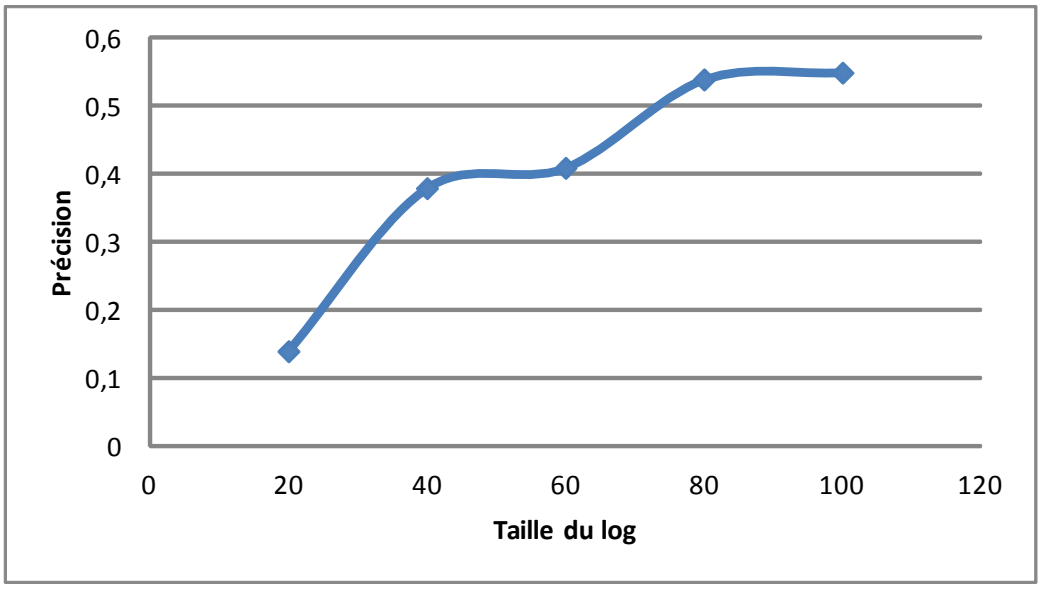


Figure 4.11 – Précision des recommandations en fonction de la taille du fichier log

état de l'art (Chapitre 2). Bien que ces travaux ont quelques similarités avec notre travail actuel, les challenges et les techniques sont très différents de ce que nous proposons. En effet, ce travail diffère de notre approche dans le sens que le système proposé *SnipSuggest* recommande les additions possibles aux différentes clauses dans la requête courante de l'utilisateur, ne complète pas la requête et la recommandation est fournie à la demande. Dans notre travail, nous proposons un système pour générer des recommandations de requêtes décisionnelles à un utilisateur. Dans notre cas, chaque requête est traitée indépendamment de n'importe qu'elle requête précédente même si elles appartiennent à la même session utilisateur. En outre, nous ne forçons pas l'utilisateur à déclarer explicitement ses préférences pour générer des recommandations puisque nous utilisons le log de requêtes au lieu du profil utilisateur.

Parmi les limites de notre approche, une est liée aux logs des requêtes : seules les requêtes présentes dans les logs peuvent être recommandées et peuvent donner lieu à des recommandations. Des problèmes peuvent aussi survenir si le système de recommandation présente une certaine faiblesse ou si les logs ne sont pas assez larges. En outre les requêtes changent dans le temps et les systèmes de recommandation de la requête doivent tenir compte de la nature dynamique des données. Enfin, le problème majeur rencontré dans ce travail est l'évaluation des recommandations. Ce problème est dû à la difficulté de définir les attributs pertinents tant que ça diffère d'un utilisateur à un autre.

4.8 Conclusion

Dans ce chapitre, nous avons posé les bases d'un système de recommandation interactive de requêtes décisionnelles dans le cadre des entrepôts de données. Nous

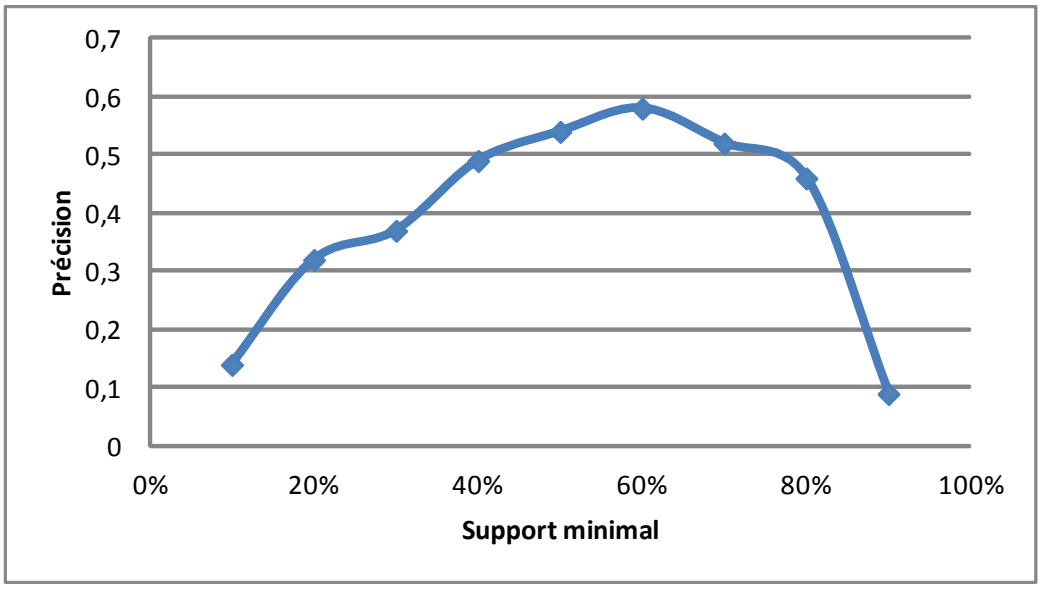


FIGURE 4.12 – Précision de recommandation en FIMIOQR en fonction de la valeur du minsup

avons explicité les principes de notre proposition, décrit le processus centré utilisateur pour permettre l'aide à l'écriture de requêtes en se basant sur l'extraction des motifs fréquents à partir des fichiers log de requêtes. Ces travaux se distinguent des travaux existants sur la recommandation de requêtes souvent basées sur l'enrichissement de requêtes utilisateur puisque notre approche aide l'utilisateur à formuler sa requête décisionnelle depuis le début tout en tenant compte de ses requêtes précédentes. L'originalité de notre méthode réside dans son aspect interactif ainsi que le recours à la méthode *Close* pour extraire les attributs (descriptifs et mesures) fréquents utilisés par les différents utilisateurs dans leurs requêtes précédentes. Pour cela, nous avons développé un système interactif de recommandation de requêtes appelé, FIMIOQR (Frequent Itemsets Mining for Interactive OLAP Query Recommendation). Notre système tient alors compte (1) des requêtes précédentes soumises à l'entrepôt de données, (2) utilise les motifs fréquents des attributs et (3) permet de proposer des attributs en adéquation avec la requête en cours d'écriture de l'utilisateur. L'approche de recommandation que nous avons proposée se base sur un processus en ligne, c'est pourquoi son efficacité et son extensibilité sont d'une importance capitale pour l'utilisateur. Une perspective directe de ce travail est de faire une veille sur les recommandations les plus acceptées par l'utilisateur afin de pouvoir les classer et les recommander en priorité à l'utilisateur dans ses sessions futures.

Chapitre 5

Recommandation de chemins de navigation

L'objectif de ce chapitre est tout d'abord de motiver la nécessité d'assister l'utilisateur durant sa navigation dans un cube de données. Nous présentons ensuite notre approche de recommandation de chemins de navigation qui s'inscrit dans la continuité de nos travaux sur la prise en compte de l'utilisateur dans les systèmes décisionnels. L'analyse OLAP, malgré son aspect interactif, ne permet pas de guider l'utilisateur vers les faits les plus pertinents pour lui lors de sa navigation au sein des cubes de données. Ceci peut s'expliquer par le fait que l'OLAP ne tient pas compte des usages, en particulier l'historique des requêtes de navigation des utilisateurs. Pour pallier ce problème qui réduit considérablement les avantages de l'analyse OLAP, nous proposons, dans ce chapitre, une approche de recommandation de chemins de navigation en s'appuyant sur l'historique des sessions d'analyse. Nous avons conçu pour cela un système de recommandation de chemins de navigation nommé NAPARE (NAvigation PAth REcommendation) qui s'appuie sur les chaînes de Markov et les sessions d'analyse précédentes des utilisateurs.

5.1 Motivation

Dans un cube de données, extrait à partir d'un entrepôt de données, les données sont organisées de manière à permettre aux décideurs de les exploiter à travers la navigation OLAP. En effet, une fois le cube OLAP initial est construit, en fonction des besoins exprimés par l'utilisateur, ce dernier peut explorer différents chemins de navigation. La navigation OLAP consiste alors en une interrogation interactive des données, en réalisant des analyses à travers de multiples passes (passant par exemple des données résumées à des données détaillées ou changeant même d'axe d'observation), successivement dans des niveaux de détail inférieurs permettant ainsi de produire les premiers résultats d'analyse pertinents pour l'aide à la décision. Ainsi, une session d'analyse typique sur un cube de données est une séquence de requêtes de navigation. Chaque requête d'une séquence est formulée sur la base des résultats de la requête précédente.

Cependant, lors de la navigation dans un cube de données, l'utilisateur n'a pas une

connaissance a priori des parties du cube susceptibles d'être intéressantes pour lui. De ce fait, choisir les prochaines navigations devient une tâche difficile pour l'utilisateur puisque plusieurs chemins de navigation se présentent à lui. Cela pourrait provoquer des temps de latence de l'analyse voire de conduire l'utilisateur dans une zone non pertinente, et de réduire ainsi les avantages de l'utilisation du système OLAP.

Le défi auquel nous nous intéressons dans ce chapitre est de pouvoir guider l'utilisateur vers les faits les plus pertinents pour lui en utilisant un système de recommandation de chemins de navigation. Généralement, la plupart des systèmes de navigation existants souffrent d'un certain nombre de limitations. Une première limitation est que le graphe de navigation est généralement très clairsemé, et nécessite du hasard ou de l'expertise des utilisateurs, ou les deux à la fois pour trouver les bons chemins menant vers les éléments recherchés. Une deuxième limitation est que les liens de navigation peuvent conduire à des résultats vides.

Toutefois, dans un cube OLAP, les données sont modélisées suivant de multiples dimensions où les sous-cubes sont généralement représentés comme des éléments de treillis de cuboïdes [HRU96]. C'est pourquoi, nous utilisons la structure de treillis de cuboïdes pour décrire les chemins de navigation d'un utilisateur. Les utilisateurs naviguent ainsi d'un cuboïde à l'autre au sein d'un cube OLAP. Le graphe de navigation est donc le treillis de tous les cuboïdes. En effet, la structure de treillis de cuboïdes permet de visualiser tous les chemins de navigation d'un utilisateur et permet surtout de garder le séquencement logique de navigation. De ce fait, nous assimilons la notion de session d'analyse à un chemin de navigation dans le treillis de cuboïdes correspondant au cube OLAP. Dans une session d'analyse, l'utilisateur passe d'une requête d'analyse à une autre qui lui est directement liée formant ainsi un enchaînement de requêtes appelé chemin de navigation. La requête d'analyse suivante ne dépend que de la requête d'analyse en cours. De ce fait, afin de guider l'utilisateur vers un chemin de navigation pertinent pour lui, notre idée est d'utiliser les chaînes de Markov qui ont la propriété suivante : "Le futur ne dépend que de l'état présent". Autrement dit, il est possible de prédire l'état suivant avec la seule connaissance de l'état présent. Ainsi, à partir d'un point de navigation (nœud courant dans le treillis de cuboïdes), et en utilisant la propriété des chaînes de Markov, il devient possible de prédire le point de navigation suivant à l'utilisateur.

5.2 Concepts généraux

Dans cette section, nous donnons les notions fondamentales utilisées dans ces travaux à savoir la navigation OLAP, le treillis de cuboïdes et le modèle de Markov.

5.2.1 Navigation dans les cubes OLAP

L'analyse OLAP donne aux utilisateurs la possibilité d'analyser et d'explorer les données de manière interactive sur la base du modèle multidimensionnel. Bien que les utilisateurs d'outils de reporting jouent essentiellement un rôle passif, les utilisateurs OLAP sont en mesure de démarrer une session d'analyse complexe où chaque étape est le résultat de l'issue de l'étape précédente. La navigation peut alors être expliquée par la transition entre les différents états [DKK05].

La navigation est un terme utilisé pour décrire le processus employé par les utilisateurs pour explorer un cube de données de façon interactive, habituellement en utilisant un client OLAP graphique connecté à un serveur OLAP. Généralement, un utilisateur commence à analyser des données en sélectionnant une requête initiale. Cette requête est constituée d'un ensemble de dimensions ainsi que d'un ensemble de conditions de filtrage. Ensuite, l'utilisateur modifie la requête de manière interactive en ajoutant ou en supprimant des colonnes (drill-down et roll-up), en ajoutant ou en supprimant les conditions de filtrage (slicing), en déplaçant des colonnes (dicing) et ainsi de suite. Nous présentons dans ce qui suit les différents opérateurs OLAP les plus utilisés dans la navigation.

- Roll-up ou forage vers le haut : consiste à représenter les données du cube à un niveau de granularité supérieur conformément à la hiérarchie définie sur la dimension.
- Drill-down ou forage vers le bas : consiste à représenter les données du cube à un niveau de granularité de niveau inférieur, donc sous une forme plus détaillée (selon la hiérarchie définie de la dimension).
- Slice : correspond à une projection selon une dimension du cube de données.
- Dice : correspond à une application d'une sélection sur plusieurs dimensions donc peut être vu comme la combinaison de plusieurs slices.
- Split ou Drill out : consiste à ajouter des dimensions, on obtient des données plus détaillées pas à cause de changement de granularité mais parce qu'on détaille en fonction de nouvelles dimensions.
- Merge ou drill in : consiste à supprimer des dimensions, on obtient des données moins détaillées par ce qu'il y a moins de dimensions mais pas de changement de granularité.
- Rotate : choisir le pivot d'analyse en faisant tourner le cube de données.

5.2.2 Treillis de cube de données

Le concept du cube de données a été proposé la première fois par Gray et al. [GBLP96] comme une généralisation de l'opérateur *Group By* de SQL pour répondre à l'enquête en ligne des données de différents points de vue des utilisateurs. Cette analyse interactive et multidimensionnelle est habituellement accomplie par agrégations pré-calculées sur les données [Sar00]. En effet, Harinarayan et al. proposent la modélisation des données dans de multiples dimensions où les vues OLAP sont généralement représentées comme des éléments de treillis [HRU96]. Le treillis de cube de données est un DAG (Directed Graph acyclique) dont les nœuds représentent des requêtes (ou cuboïdes) qui sont caractérisées par les attributs de la clause Group By. Les arêtes dénotent la relation de dérivabilité entre les cuboïdes. Par exemple, un cube de données avec des dimensions A_1, A_2, A_3 est présenté dans la Figure 5.1 (a) comme une structure en treillis. Généralement, dans le contexte OLAP, les dimensions sont organisées en hiérarchies de dimensions, qui peuvent être également représentées par un treillis. Par exemple, la Figure 5.1 (b) montre le treillis des attributs de trois dimensions A_1, A_2, A_3 où a_{ij} est le $j^{\text{ème}}$ niveau dans la hiérarchie de

la dimension A_i. L'élément supérieur de chaque treillis est "all", ce qui signifie qu'il n'y a pas de regroupement pour cette dimension.

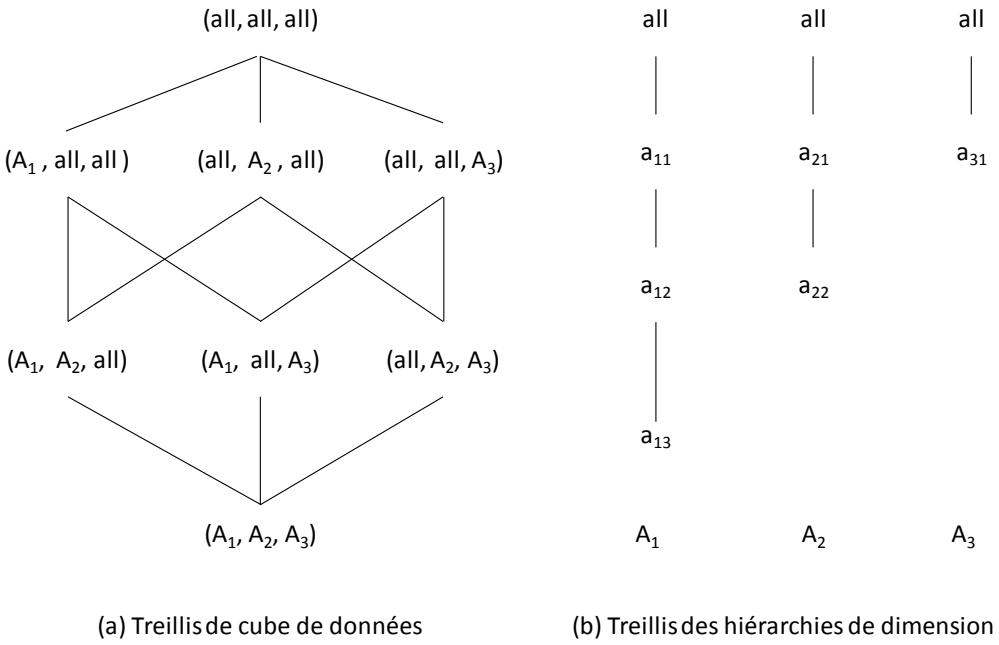


Figure 5.1 – Exemples de treillis de cubes OLAP

Nous pouvons construire le treillis impliquant les hiérarchies qui représente l'ensemble des points de vue qui peuvent être obtenus en regroupant sur chaque combinaison d'éléments l'ensemble des hiérarchies de dimension. La Figure 5.2 montre le treillis qui combine le treillis de cube de données de la figure 5.1 (a) avec les treillis de la hiérarchie des dimensions de la Figure 5.1 (b).

Par conséquent, le treillis de cube de données avec les hiérarchies de dimensions fournit un moyen d'agréger les données selon plusieurs niveaux de granularité qui fournissent un moyen intuitif pour les analystes afin de naviguer à différents niveaux de détail de l'information. Toutefois, la hiérarchie introduit un problème fondamental de calcul de cube efficace : le nombre de cuboïdes dans un treillis augmente avec le nombre de dimensions ainsi que le nombre de niveaux de hiérarchies. Ainsi, le nombre total de chemins dépend du nombre des nœuds dans le treillis. Sachant que n_i est le nombre de niveaux des hiérarchies, selon Caron et al. [CD08], le nombre de chemins dans un treillis est donné par la formule suivante.

$$\text{nombre de chemins} = \frac{(n_1 + n_2 + ... + n_k)!}{n_1!n_2!...n_k!}$$

Dans notre exemple dans la Figure 5.2, nous avons 3 dimensions avec respectivement trois, deux et un niveaux de hiérarchies. Nous avons alors $\frac{(3+2+1)!}{3!*2!*1!} = \frac{720}{12} = 60$ chemins différents. Nous constatons qu'avec seulement 3 dimensions et 6 niveaux de hiérarchies, on obtient 60 chemins différents. Ainsi, le nombre de chemins d'analyse augmente avec, à la fois, le nombre de dimensions et le nombre des niveaux d'analyse.

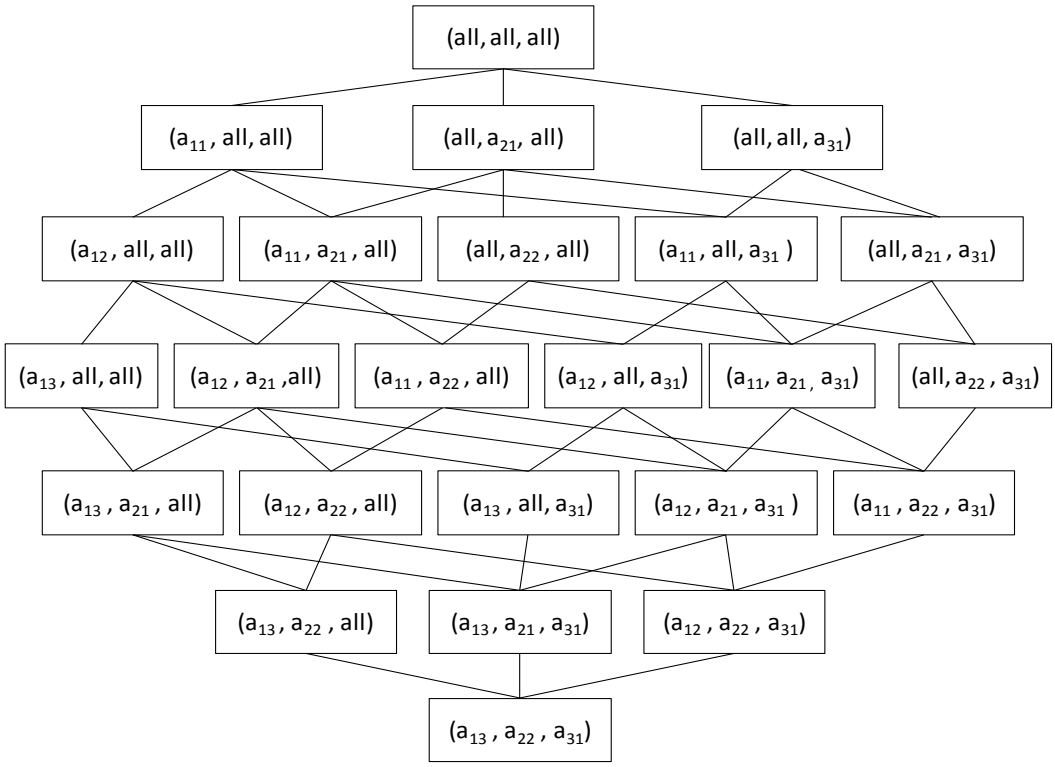


FIGURE 5.2 – Treillis combiné

Définition 14. Ordre partiel dans un treillis de cuboïdes. L'arête entre deux cuboïdes c et c' représente la relation de dépendance entre ces deux cuboïdes. On dit que c est dépendant de c', noté $c \preceq c'$, si une requête répondue par c peut être aussi répondue par c', mais l'inverse n'est pas vrai. On dit aussi que $c \preceq c'$ si et seulement si c peut être calculé à partir de c'.

Notons que $\preceq$ impose un ordre partiel sur les nœuds et qu'il est transitif. Pour un ensemble d'éléments d'un treillis, deux éléments doivent avoir une borne supérieure et une borne inférieure selon l'ordre partiel $\preceq$. Cependant, dans la pratique, nous avons seulement besoin des hypothèses suivantes : (a) $\preceq$ est un un ordre partiel, et (b) il existe un élément supérieur, un nœud duquel chaque nœud dépend.

Par exemple, dans le treillis de la Figure 5.1, supposons que *Magasin*, *Produit* et *Date* correspondent respectivement à A_1, A_2 et A_3 et considérons les deux cuboïdes (all, A_2, all) et (all, A_2, A_3). Si nous voulons connaître les ventes de chaque produit, il est évident que nous ne pouvons pas répondre à cette question en utilisant le cuboïde (all, A_2, all) ou (all, A_2, A_3) suivants. Mais si nous voulons connaître les ventes de chaque produit à des dates différentes, nous pouvons seulement obtenir la réponse à partir du cuboïde (all, A_2, all).

Par ailleurs, il y a certains cuboïdes qui ne sont pas comparables en utilisant l'opérateur $\preceq$. Par exemple, le cuboïde (A_1, all, all) et le cuboïde (all, all, all) ne sont pas comparables.

Définition 15. Ancêtres et descendants. Étant donné un fait avec N dimensions, il existe 2^N cuboïdes. Nous définissons les ancêtres et les descendants d'un cuboïde c de la manière suivante :

$$Anc(c) = \{c' \mid c \preceq c'\}$$
$$Des(c) = \{c' \mid c' \preceq c\}$$

Définition 16. Parents/Enfants. Le parent/enfant d'un nœud c dans un treillis est défini comme l'ancêtre/descendant immédiat de c. Le parent P (ou enfant E) de c peut être défini comme suit :

$$P(c) = \{c' \mid c \preceq c', \nexists x, c \preceq x, x \preceq c'\}$$
$$E(c) = \{c' \mid c' \preceq c, \nexists x, c' \preceq x, x \preceq c\}$$

Par exemple dans la Figure 5.1, les ancêtres de cuboïde (A_1, all, all) sont les cuboïdes (A_1, A_2, all), (A_1, all, A_3) et (A_1, A_2, A_3) tandis que son parent est le cuboïde (A_1, all, A_3).

Définition 17. Chemin de navigation dans un cube OLAP. Un chemin de navigation dans un cube de données est défini par une séquence ordonnée de requêtes $q_1, q_2, ..., q_n$, appliquées aux cuboïdes $c_1, c_2, ..., c_n$, respectivement, tels que $c_1 \diamond c_2, c_2 \diamond c_3, ..., c_{n-1} \diamond c_n$ où $\diamond$ est soit $\preceq$ soit $\succeq$. Autrement dit, étant donnée une requête, un utilisateur applique l'opérateur drill-down/roll-up pour passer d'un niveau supérieur à un niveau inférieur et vice-versa.

5.2.3 Modèle de Markov

Les chaînes de Markov [Mar71] modélisent des relations entre des éléments dans le temps selon une hypothèse d'indépendance telle que la probabilité d'apparition d'un élément n'est dépendante que de l'élément précédent. Un modèle de Markov correspond à un simple graphe d'états, doté d'une fonction de transition probabiliste. Une chaîne de Markov désigne les processus qui commencent dans l'un de ces états, et évoluent d'état en état selon les transitions.

A chaque pas de temps, le modèle subit une transition qui va potentiellement modifier son état. Cette transition permet donc au système modélisé d'évoluer, selon une loi connue par avance. Néanmoins, cette loi de transition est probabiliste. En effet, l'évolution du système peut être incertaine, ou simplement mal connue. Cette fonction probabiliste permet donc d'exprimer simplement la loi d'évolution du modèle, sous la forme d'une matrice de probabilités. Cela ouvre donc la porte à un très grand nombre d'utilisations où l'évolution d'un système n'est connue qu'à travers des statistiques.

Un modèle de Markov peut être décrit de la manière suivante. Étant donné un ensemble d'états $S = \{s_1, s_2..., s_n\}$, le processus débute dans un de ces états et se propage successivement d'un état à l'autre. Chaque mouvement correspond à une étape du processus. Si le processus se trouve dans l'état s_i alors il peut se trouver dans l'état suivant s_j avec une probabilité p_{ij}. Cette probabilité ne dépend pas du chemin effectué depuis le début du processus mais uniquement de l'état s_i. L'ensemble des probabilités p_{ij}, $1 < i, j < n$, n étant le nombre total d'états composant le modèle, est représenté par la matrice de transitions notée MT. Notons également que le processus peut rester dans le même état d'une étape à l'autre avec une probabilité p_{ii}.
Pour définir totalement le modèle, il reste à définir un vecteur $u = (u_1, u_2, ...u_n)$ correspondant aux probabilités de débuter le processus dans l'un ou l'autre des états. La probabilité de débuter dans l'état s_i est u_i. L'état s_i est appelé alors état initial. Un modèle de Markov doit respecter les deux contraintes suivantes :

- La somme des probabilités des états initiaux est égale à 1 :$\sum_{i=1}^{n} u_i = 1$.
- La somme des probabilités des transitions partant d'un état est égal à 1 : $\forall i, 1 \leq i \leq n, \sum_{j=1}^{n} p_{ij} = 1$.

5.3 Recommandation collaborative de chemins de navigation

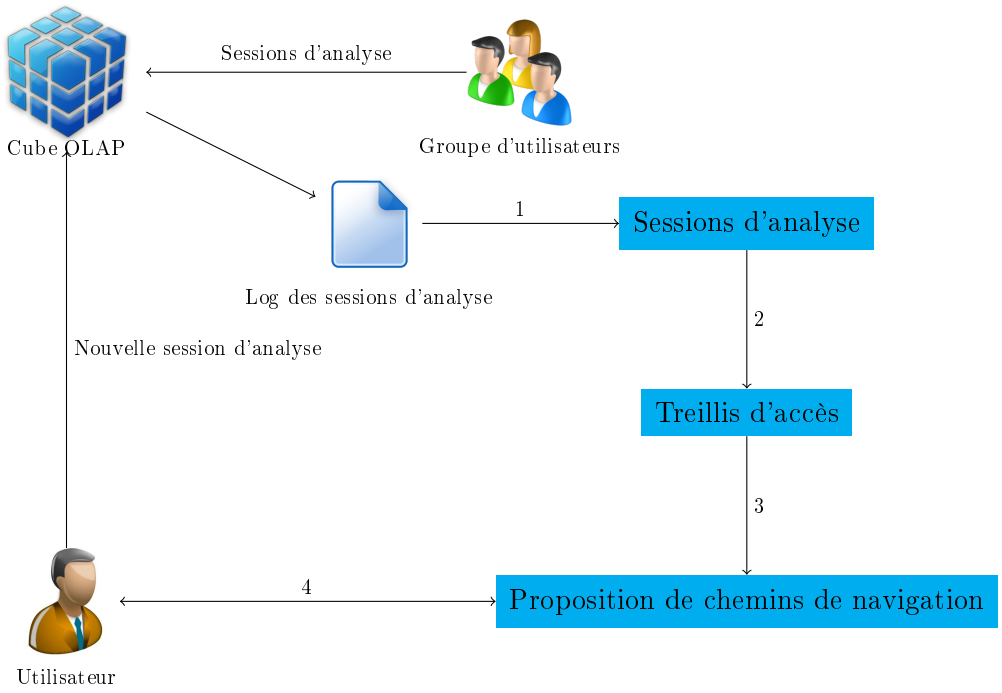
5.3.1 Principe

Dans la littérature, l'élément principal à recommander dans les systèmes de recommandation, dans le domaine des bases de données ou des entrepôts de données, est la requête. Concrètement, dans les systèmes décisionnels, de nombreux efforts ont été déployés pour se concentrer sur l'interrogation d'un entrepôt de données, du comportement de navigation sans tenir compte du comportement de l'utilisateur lors de sa phase de navigation dans les cubes OLAP. Les utilisateurs sont souvent livrés à eux mêmes et ne sont pas guidés dans le processus de navigation.

Pour pallier ce problème, nous développons une approche centrée utilisateur qui suggère à l'utilisateur des chemins de navigation. L'idée est de réaliser une navigation efficace en fournissant des conseils de navigation. Afin de recommander des chemins de navigation pertinents, nous utilisons les usages (l'historique des accès) que nous extrayons du log de sessions d'analyse. Étant donné qu'un cube de données peut être considéré comme un treillis de cuboïdes, notre approche tire profit de cette structure pour construire un modèle probabiliste dans lequel sont stockées des estimations de probabilités. Ces probabilités peuvent alors être utilisées pour calculer le chemin de navigation le plus probable. Nous utilisons le modèle Markov pour modéliser le comportement de l'utilisateur lors de sa navigation dans un cube OLAP parce qu'il offre un moyen simple pour saisir la dépendance séquentielle entre un niveau d'analyse (état) à l'autre (nouvel état). En effet, l'adéquation des modèles de Markov

pour la navigation a été étudiée et reconnue dans la littérature depuis plusieurs années pour prédire la requête de l'utilisateur dans le Web [ZHH02, EVK05, SN06].

Dans ce cadre, notre approche consiste à prédire le ou les prochains cuboïdes qu'un utilisateur donné est susceptible de visiter ou consulter. Ainsi, il est possible d'effectuer de la recommandation à partir de ces prédictions, simplement en recommandant les navigations prédites ayant la plus grande probabilité.



1. Extraction des sessions d'analyse à partir du fichier log des utilisateurs
2. Construction de treillis d'accès (treillis de cuboïdes + probabilités de transition entre les nœuds)
3. Génération de recommandations de chemins de navigation
4. Dialogue entre l'utilisateur et le système de recommandation

Figure 5.3 – Processus général de recommandation de chemins de navigation

Notre approche de recommandation va rendre la navigation dans les cubes OLAP plus intéressante pour l'utilisateur puisqu'elle permet de le guider vers les chemins les plus pertinents pour lui. En effet, notre approche guide l'utilisateur à atteindre son objectif d'analyse en moins d'étapes de navigation et surtout elle lui empêche de tourner au rond sur le cube de données et de se perdre dans les données multidimensionnelles.

Notre approche de recommandation de chemins de navigation se déroule en trois phases. (1) Tout d'abord, à partir d'un ensemble de sessions d'analyses issues des fichiers logs des utilisateurs, un prétraitement est nécessaire pour extraire les différentes sessions d'analyse. (2) Nous procédons ensuite à construire le treillis d'accès (treillis de cuboïdes + probabilités de transition entre les nœuds en fonction de l'historique de navigation) appelé plus communément DAL (Data cube Access Lattice). (3) Nous exploitons le treillis d'accès pour recommander à l'utilisateur des chemins de navigation et l'aider à anticiper dans la navigation pour aller vers les faits les plus

pertinents. Cette phase est elle-même effectuée en deux étapes : (i) construction de la liste des recommandations candidates en appliquant le modèle de Markov sur les sessions extraites, et (ii) ordonnancement de ces recommandations candidates. Ainsi, notre approche de recommandation de chemins de navigation fournit à l'utilisateur un ensemble de recommandations et, en même temps, met à jour le treillis d'accès.

L'approche que nous proposons s'intègre parfaitement dans le processus habituel d'interrogation des entrepôts de données. Cette intégration est non intrusive et l'utilisateur peut décider d'accepter ou de ne pas accepter les suggestions du système. Le processus général que nous proposons pour notre approche est illustré dans la Figure 5.3. Par ailleurs, la particularité de nos travaux sur la recommandation de chemins de navigation réside dans son interactivité. La recommandation se fait au fur et à mesure de la navigation de l'utilisateur au sein du cube OLAP.

5.3.2 Processus de recommandation de chemins de navigation

Nous présentons dans cette section les différentes étapes de notre approche de recommandation de chemins de navigation basée sur les chaînes de Markov.

Étape 1. Prétraitement

Notre approche de recommandation de chemins de navigation est basée sur le log de sessions de l'ensemble des utilisateurs. Cependant, ce dernier peut être très volumineux vu le nombre d'utilisateurs et le nombre de sessions d'analyse possibles. Par ailleurs, les utilisateurs peuvent avoir différents intérêts et ainsi ils peuvent naviguer dans des parties différentes du cube OLAP. Par conséquent, la première étape de notre stratégie de recommandation consiste à prétraiter le log des requêtes d'analyse. En effet, les tâches de prétraitement, y compris le nettoyage des données, l'identification de l'utilisateur et l'identification de sessions peuvent être appliquées au log de sessions pour obtenir toutes les sessions d'analyse.

Le log peut être divisé en sessions par de nombreuses façons telles que l'identificateur utilisateur (user id), la date (timestamp) et la durée d'une session. Dans nos travaux, nous combinons le nom d'utilisateur et le seuil de délai d'attente (timeout treshold) pour délimiter les sessions d'analyse. Le seuil de délai d'attente est fixé à moins de 12 heures dans le système SQL Server. Ainsi, le log de sessions est divisé en plusieurs parties tout d'abord par l'identificateur de l'utilisateur, puis nous comparons le timestamp de deux requêtes consécutives. Si le délai d'attente entre ces deux requêtes dépasse 12 heures, la requête ayant le plus petit timestamp représente un point de départ pour une nouvelle session d'analyse. Cette étape nous permet d'obtenir des sessions (des séquences de requêtes) comme illustré dans la Figure 5.3.

Le Tableau 5.1 présente un exemple d'identification de différentes sessions d'analyse dans un log. La première colonne représente une collection de sessions de navigation avec l'état de départ et l'état final. La deuxième colonne du Tableau 5.1 représente la fréquence de la session d'analyse qui représente le nombre de fois où la séquence correspondante de requêtes est traversée ou visitée dans le log.

Session	Fréquence
$s_1 \rightarrow s_2 \rightarrow s_4 \rightarrow s_7$	3
$s_1 \rightarrow s_2 \rightarrow s_4 \rightarrow s_5 \rightarrow s_7$	2
$s_1 \rightarrow s_3 \rightarrow s_4 \rightarrow s_7$	4
$s_1 \rightarrow s_2 \rightarrow s_4 \rightarrow s_6 \rightarrow s_7$	2
$s_1 \rightarrow s_4 \rightarrow s_7$	2
$s_1 \rightarrow s_4 \rightarrow s_5 \rightarrow s_7$	1
$s_1 \rightarrow s_4 \rightarrow s_6 \rightarrow s_7$	2
$s_1 \rightarrow s_2 \rightarrow s_5 \rightarrow s_7$	3
$s_5 \rightarrow s_4 \rightarrow s_2$	1
$s_6 \rightarrow s_4 \rightarrow s_3$	1
$s_4 \rightarrow s_2$	2

Tableau 5.1 – Sessions d'analyse identifiées dans le log

Étape 2. Construction de treillis d'accès

La deuxième étape de notre stratégie de recommandation de chemins de navigation consiste à construire le treillis d'accès. En fait un treillis d'accès est le treillis de cuboïdes avec les probabilités de transition entre les nœuds en fonction de l'historique de navigation. Une fois, les différentes sessions d'analyse obtenues, elles peuvent correspondre à un graphe pondéré appelé modèle de Markov. Le modèle de Markov se compose des états qui représentent les requêtes des utilisateurs et un lien ou arête entre deux états successifs. Chaque état ou cuboïde est défini par une identité ($s_1, s_2, etc.$). Chaque lien ou arête est désigné par un certain nombre de visites des séquences de requêtes. La chaîne de Markov est définie par un ensemble d'états avec leurs probabilités initiales et une matrice de transition MT. L'ensemble des états est constituée par l'état initial, l'état final et les états intermédiaires qui correspondent aux cubïdes visités. Dans ce contexte, nous proposons de modéliser le cube de données comme une chaîne de Markov où les cuboïdes correspondent aux états et le chemin de navigation correspond aux transitions entre les états. Les chemins de la chaîne ayant la probabilité la plus élevée seront les chemins préférés, les plus probables sur le cube OLAP. Pour cela, nous calculons les différentes probabilités en se basant sur le modèle de Markov. Une fois le treillis d'accès est construit, nous pouvons l'exploiter pour recommander des chemins de navigation à l'utilisateur. Étant donnée une requête courante de l'utilisateur, nous identifions à quel état elle correspond dans le modèle de Markov. A partir de cet état et en utilisant la matrice de transitions, nous pouvons calculer plusieurs probabilités de passer de l'état présent à un autre état.

Par conséquent, nous utilisons une approche modèle. L'avantage est que ce modèle peut être construit hors ligne à partir de l'historique des utilisateurs et être utilisé rapidement en ligne pour calculer les recommandations. Par conséquent, la problématique de la complexité en temps de la construction de ce modèle peut être secondaire. Dans le cadre de notre approche, la prédiction de chemin de navigation peut être calculée par la construction d'un modèle probabiliste dans lequel sont stockées des estimations de probabilités. Ces probabilités peuvent alors être utilisées

pour calculer le chemin de navigation le plus probable. La problématique est alors de construire un modèle à partir duquel il sera possible d'obtenir les probabilités. Ainsi, nous construisons le modèle de Markov sous la forme de treillis de cuboïdes afin de prédire les navigations prochaines en se basant sur les visites des utilisateurs.

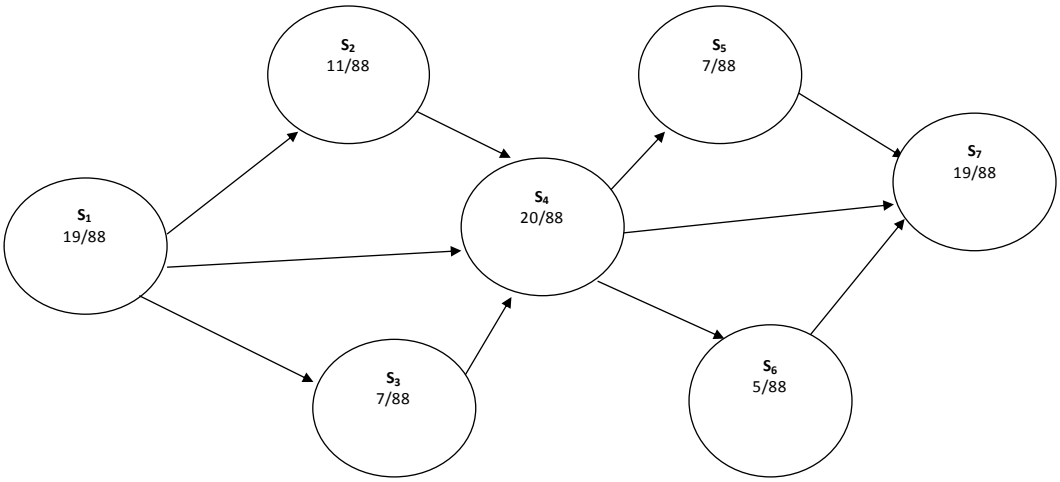


FIGURE 5.4 – Modèle de Markov des sessions d'analyse

La Figure 5.4 représente le modèle de sessions données dans le Tableau 5.1. Chaque état a une identité et une probabilité d'état.

Probabilité d'état La probabilité d'état P(s) (state probability) représente la probabilité de soumettre une requête. Cette dernière correspond à un cuboïde. La probabilité d'état peut être calculée par la formule suivante :

$$P(s) = \frac{N_{s_i}}{N}$$

où

- N_{s_i} : nombre de fois où la requête s_i a été soumise.
- N : nombre total de requêtes dans le log des utilisateurs.

Par exemple, à partir du Tableau 5.1, nous pouvons calculer la probabilité de l'état s_2 celle-ci est égale à 11/88 car la requête s_2 a été visitée 11 fois et que le nombre total de requêtes de log de l'utilisateur est 88.

Probabilité de transition Chaque lien entre les nœuds d'un treillis représente la probabilité de transition (TP) qui est calculée selon le nombre de fois où le lien correspondant est suivi ou visité au nombre de fois que le nœud d'ancrage a été visité. La probabilité de transition est représentée dans la matrice de transitions MT qui enregistre les probabilités de transition. Ainsi, la probabilité de transition peut être calculée par la formule suivante :

$$TP(s_i \to s_j) = \frac{N_{(s_i \to s_j)}}{N_{s_i}}$$

où $N(s_i \rightarrow s_j)$ est le nombre de fois où le lien entre s_i et s_j a été emprunté. Par conséquent, notre treillis d'accès est représenté par la matrice de transitions MT qui peuvent être calculées à partir des probabilités des états. A partir de cette matrice MT et la position courante (état actuel), nous allons calculer les probabilités de tous les chemins de navigation possibles dans le graphe de Markov commençant par cet état courant.

Par exemple, nous pouvons calculer la probabilité de transition de s_2 à s_4 comme suit : $TP(s_2 \rightarrow s_4) = 5/11$ et la probabilité de transition s_4 à s_6 comme suit : $TP(s_4 \rightarrow s_6) = 4/20$. De plus, nous pouvons calculer la probabilité de transition dans le sens contraire comme par exemple la probabilité de transition de s_4 à s_2 qui est égale à $TP(s_4 \rightarrow s_2) = 3/20$. Le Tableau 5.2 montre la matrice de transitions MT qui représente toutes les probabilités de transition possibles.

MT	s_1	s_2	s_3	s_4	s_5	s_6	s_7
s_1	0	8/19	6/19	5/19	0	0	0
s_2	3/11	0	0	5/11	3/11	0	0
s_3	0	0	0	6/7	1/7	0	0
s_4	0	3/20	1/20	0	3/20	4/20	9/20
s_5	0	0	0	1/7	0	0	6/7
s_6	0	0	0	1/5	0	0	4/5
s_7	0	0	0	0	0	0	0

Tableau 5.2 – Matrice de transitions

Probabilité de chemin Un chemin est une séquence finie d'états qui sont accessibles dans l'ordre de leur parcours dans le cube de données sous-jacent. Dans ce contexte, la probabilité d'un chemin de navigation est estimée par le produit de la probabilité initiale du premier état du chemin de navigation et les probabilités des transitions. La règle de la chaîne de Markov est appliquée afin de calculer toutes les probabilités de chemin. Ainsi, la probabilité de chemin de navigation PP peut être calculée par la formule suivante :

$$PP(s_i \rightarrow s_j) = P(s_i \prod TP(s_i \rightarrow s_j))$$

Par exemple, la probabilité estimée du chemin $(s_1 \rightarrow s_3 \rightarrow s_4 \rightarrow s_7)$ est égale à : $19/88 * 6/19 * 6/7 * 9/20$. Le Tableau 5.3 montre les probabilités de chemins des sessions données dans le log comme des exemples de calcul de probabilités de chemins.

Session	Fréquence	Probabilité de chemin
$s_1 \rightarrow s_2 \rightarrow s_4 \rightarrow s_7$	3	19/88 * 8/19 * 5/11 * 9/20
$s_1 \rightarrow s_2 \rightarrow s_4 \rightarrow s_5 \rightarrow s_7$	2	19/88 * 8/19 * 5/11 * 3/20 * 6/7
$s_1 \rightarrow s_3 \rightarrow s_4 \rightarrow s_7$	4	19/88 * 6/19 * 6/7 * 9/20
$s_1 \rightarrow s_2 \rightarrow s_4 \rightarrow s_6 \rightarrow s_7$	2	19/88 * 6/19 * 6/7 * 4/20 * 4/5
$s_1 \rightarrow s_4 \rightarrow s_7$	2	19/88 * 5/19 * 9/20
$s_1 \rightarrow s_4 \rightarrow s_5 \rightarrow s_7$	1	19/88 * 5/19 * 4/20 * 4/5
$s_1 \rightarrow s_4 \rightarrow s_6 \rightarrow s_7$	2	19/88 * 5/19 * 4/20 * 4/5
$s_1 \rightarrow s_2 \rightarrow s_5 \rightarrow s_7$	3	7/88 * 1/7 * 3/20
$s_5 \rightarrow s_4 \rightarrow s_2$	1	5/88 * 1/5 * 1/20
$s_6 \rightarrow s_4 \rightarrow s_3$	1	20/88 * 3/20
$s_4 \rightarrow s_2$	2	3/20

Tableau 5.3 – Probabilités de chemins de navigation

Quand l'utilisateur navigue dans le cube de données, nous lui recommandons à chaque étape la meilleure voie qui peut prendre. Ainsi, le meilleur chemin de navigation sera calculé au fur et à mesure de la navigation de l'utilisateur.

Étape 3. Recommandation collaborative de chemins de navigation

Les systèmes de recommandation sont généralement classés selon deux catégories : des systèmes basés sur le contenu et ceux basés sur le filtrage collaboratif [AT05]. Les systèmes basés sur le contenu recommandent à l'utilisateur des éléments similaires à ceux qui l'ont intéressé dans le passé, tandis que les systèmes basés sur le filtrage collaboratif recommandent des éléments qui ont intéressé des utilisateurs qui lui sont similaires. Ce travail s'inscrit dans une approche de recommandation collaborative impliquant plusieurs utilisateurs parce que nous exploitons les sessions d'analyse de l'ensemble des utilisateurs du système décisionnel. L'idée d'exploiter ce que les autres utilisateurs ont fait pour produire des recommandations est très populaire dans le domaine de la recherche d'informations [AT05], et dans l'exploitation des usages du Web (Web Usage Mining) [SCDT00]. Notre contribution est d'adapter ces techniques existantes à l'OLAP. Généralement, dans la recommandation collaborative, le problème de base est le suivant : on dispose d'un ensemble de m éléments (livres, films, objets, etc.), de n utilisateurs et d'une matrice d'utilité $R = (r_{ij}, i = 1, ...n, j = 1, ...m)$ (utility matrix ou bien rating matrix) telle que :

- $r_{ij} \in R$ signifie que l'utilisateur i a attribué la note r_{ij} à l'élément j ;
- $r_{ij} = *$ signifie que la note de l'utilisateur i à l'élément j n'est pas connue.

Parallèlement, dans nos travaux, la matrice d'utilité pour notre système de recommandation de chemins de navigation n'est que la matrice de transitions MT qui représente le treillis d'accès.

Algorithme 11 : NAPARE(MT, NC, CA)

Entrées :
MT : Matrice de transitions
NC : Nœud courant
CA : Condition d'arrêt
Sortie :
Ch : Chemin de navigation à recommander
Début
  ajouter NC au Ch ;
  Tant que $\neg$ *vide(Ch)* **faire**
    c = premier élément de Ch ;
    Pour chaque *a dans* $Parent(NC) \cap C$ *ou a dans* $Enfant(NC) \cap C$ **faire**
      Si *((*$\neg$ *CA) et Max(TP[NC][a]))* **alors**
        ajouter a à la fin de Ch ;
      Finsi
    Finpour
  fintantque
  Retourner *Chemin de navigation* Ch ;
  `// Retourner le chemin de navigation le plus pertinent`
Fin

Par conséquent, à partir de l'état actuel de l'utilisateur dans le treillis d'accès et la matrice de transitions MT, nous recommandons à l'utilisateur le chemin de navigation le plus pertinent à emprunter. Nos recommandations sont obtenues à partir de la matrice de transitions. Ceci peut être expliqué par l'algorithme de recommandation de chemins de navigation (Algorithme 11) qui calcule des recommandations candidates en se basant sur la requête courante et l'historique de l'utilisateur représentée par la matrice de transitions. Les notations utilisées dans l'algorithme sont les suivantes :

- Ch désigne la file d'attente de nœuds (cuboïdes dans le treillis).
- c est le premier nœud de Ch.
- C est l'ensemble des cuboïdes dans le treillis.
- NC est le nœud courant qui représente le début d'une session d'analyse.
- $TP[i][j]$ désigne la probabilité d'un arête reliant les nœds i et j dans le treillis de cuboïdes. C'est la probabilité que la prochaine requête sera adressée au cuboïde j sachant que la requête courante est adressée au cuboïde i. Le log de sessions d'analyse est utilisé pour déterminer les probabilités des arêtes. Ces valeurs sont mises à jour entre deux sessions successives.
- CA désigne la condition d'arrêt fixée par l'utilisateur. Elle peut être présentée par une longueur de chemin ou une valeur de seuil de probabilité exigées par l'utilisateur.

Notre système de recommandation NAPARE utilise comme base le treillis d'accès et la requête courante de l'utilisateur. Tout d'abord, le modèle de Markov est construit

à partir de treillis de cubïdes et l'historique des utilisateurs. Le résultat est le treillis d'accès (DAL) qui permet de résumer le comportement interrogatoire des utilisateurs passés. Il est représenté par la matrice de transitions (MT).

L'algorithme commence avec la première requête de l'utilisateur qui représente le nœud courant ou l'état actuel. L'algorithme effectue une première recherche étendue pour calculer la probabilité de chaque nœud. La condition d'arrêt peut être la longueur de chemin ou une valeur de seuil appliquée à la probabilité de l'arête. Par exemple, si la probabilité d'un nœud est inférieure à la valeur de seuil ou si la longueur du chemin de navigation dépasse une certaine valeur, alors on pourrait arrêter la poursuite de nœuds le long de ce chemin. Ensuite, NAPARE cherche des correspondances entre la session active de l'utilisateur et les états du modèle (différents nœuds du treillis d'accès). Enfin, le chemin le plus probable sera recommandé à l'utilisateur.

5.4 Application de NAPARE sur un cube OLAP

Prenons l'entrepôt de données Foodmart décrit dans le Chapitre 4 comme exemple pour y appliquer notre système de recommandation NAPARE. Pour de raison de clarté, nous prenons juste le cube de données *Sales* qui comprend la mesure *Store Sales* et les dimensions *Store*, *Time* et *Product*. En effet, le fait *Sales* présente les ventes d'un produit dans un magasin à un moment donné avec la mesure *Store Sales* comme présenté dans la Figure 5.5. La dimension *Time* possède trois niveaux de hiérarchies *Time*, *Month* et *Year* ainsi que la dimension *Store* est hiérarchisée selon deux niveaux *Store* et *City*.

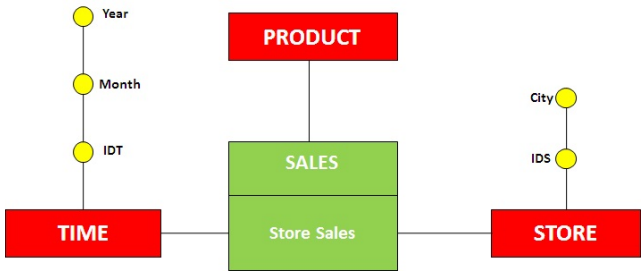


FIGURE 5.5 – Schéma de cube de données *Sales*

Comme expliqué dans la section 5.2.1, nous pouvons présenter le cube de données sous forme de treillis de cuboïdes. Ainsi, le treillis de cuboïdes correspondant au cube de données *Sales* (présenté dans la Figure 5.5) est donné par le treillis présenté dans la Figure 5.6.

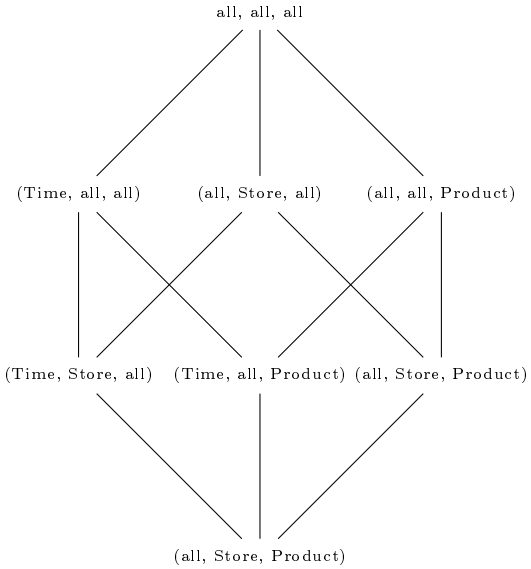


Figure 5.6 – Treillis de cuboïdes de cube de données *Sales*

Nous combinons le treillis de cuboïdes avec les hiérarchies de dimensions. Le résultat est présenté dans la Figure 5.7. C'est un treillis qui comporte 24 nœuds qui représentent tous les cuboïdes du cube OLAP.

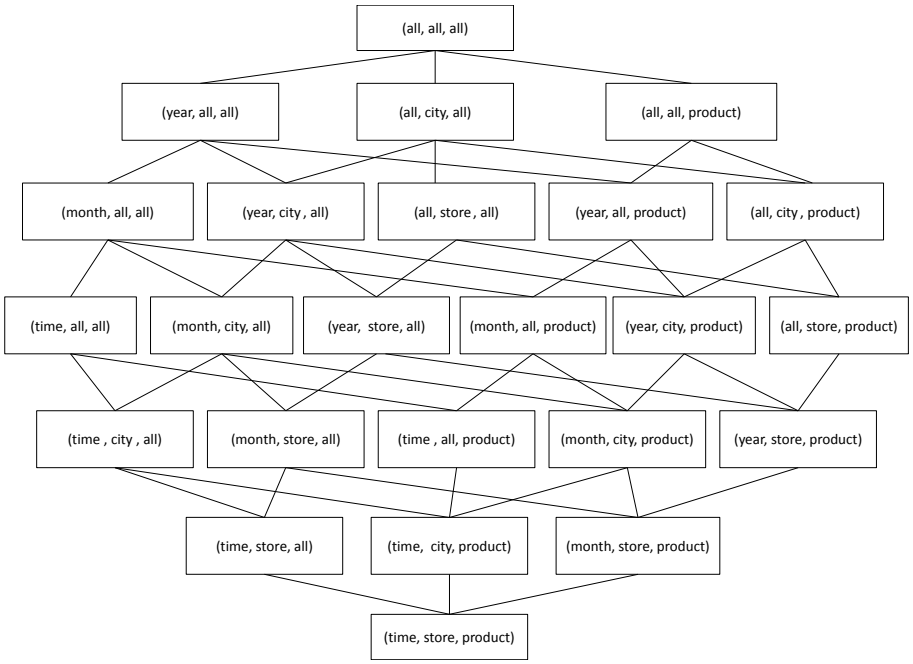


Figure 5.7 – Treillis de cuboïdes impliquant les hiérarchies de dimensions

5.4.1 Construction du treillis d'accès

Nous utilisons le modèle de chaîne de Markov [Mar71] pour prédire l'interaction de l'utilisateur avec le cube OLAP et pour la modélisation de la navigation dans un cube de données. Chaque cuboïde visité (accessible) par l'utilisateur correspond à un état de la chaîne de Markov. La probabilité que l'utilisateur suit un chemin de navigation particulier correspond alors aux probabilités de transition d'état associées à une chaîne de Markov. Ainsi, à partir des visites des utilisateurs, nous calculons les probabilités d'états et de transitions comme expliqué dans la Section 4.5. Le treillis d'accès est présenté par une matrice de transitions. Dans notre cas, la matrice de transitions est égale à la matrice d'adjacence du treillis avec les probabilités de transitions entre les arêtes. C'est une matrice carrée symétrique (graphe non orienté) d'ordre 24 (nombre de nœuds du treillis de cuboïdes).

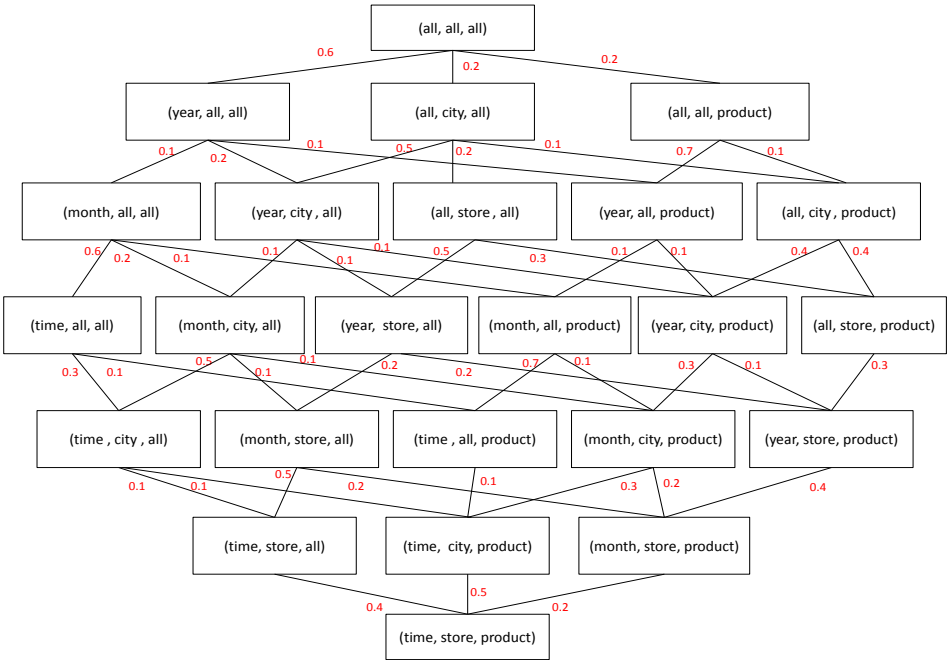


FIGURE 5.8 – Treillis d'accès

5.4.2 Recommandation de chemins de navigation

La probabilité qu'un utilisateur applique l'opérateur drill-down à un cuboïde particulier c' étant donné qu'il interroge actuellement un cuboïde c peut être trouvée en calculant la probabilité de transition entre les deux nœuds dans la matrice de probabilité de nième degré où n est égal au nombre de sauts (longueur du chemin) pour atteindre l'état (cuboïde) c' à partir de l'état (cuboïde) c.

Supposons qu'un utilisateur commence son analyse à partir du nœud NC (all, all, all) qui correspond à sa première requête. Ayant la matrice de transitions, nous

y appliquons l'algorithme NAPARE (Algorithme 11). La condition d'arrêt est la longueur du chemin qui est fixée par l'utilisateur à 3. Ainsi, en se basant sur l'algorithme NAPARE, nous avons NAPARE(MT, (all, all, all), n=3) avec MT est la matrice de transitions présentée dans le Tableau 5.4.

Par conséquent, les recommandations candidates de chemins de navigation sont les chemins ayant la probabilité maximale. Dans notre cas, 3 chemins se présentent ayant la même probabilité qui est égale à $0.6 * 0.2 * 0.1 = 0.072$.

- Chemin 1 : $(all, all, all) \rightarrow (year, city, all) \rightarrow (month, city, all)$
- Chemin 2 : $(all, all, all) \rightarrow (year, city, all) \rightarrow (year, store, all)$
- Chemin 3 : $(all, all, all) \rightarrow (year, city, all) \rightarrow (year, store, product)$

Si l'utilisateur ne choisit aucune de ces suggestions et accède au cuboïde $(all, all, product)$, NAPARE trouve les chemins candidats suivants :

- Chemin 1 : $(year, all, product) \rightarrow (month, all, product) \rightarrow (time, all, product)$ ayant la probabilité $0.7 * 0.1 * 0.7 = 0.049$.
- Chemin 2 : $(year, all, product) \rightarrow (year, city, product) \rightarrow (month, city, product)$ avec la probabilité $0.7 * 0.1 * 0.3 = 0.021$.

NAPARE recommande à l'utilisateur le chemin 1 parce qu'il présente le chemin le plus probable avec une probabilité égale à 0.049.

MT	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24
1	0	0.6	0.2	0.2	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
2	0.6	0	0	0	0.1	0.2	0	0.1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
3	0.2	0	0	0	0	0.5	0.2	0	0.1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
4	0.2	0	0	0	0	0	0	0.7	0.1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
5	0	0.1	0	0	0	0	0	0	0	0.6	0.2	0	0.1	0	0	0	0	0	0	0	0	0	0	0
6	0	0.2	0.5	0	0	0	0	0	0	0	0.1	0.1	0	0.1	0	0	0	0	0	0	0	0	0	0
7	0	0	0.2	0	0	0	0	0	0	0	0	0.5	0	0	0.3	0	0	0	0	0	0	0	0	0
8	0	0.1	0	0.7	0	0	0	0	0	0	0	0	0.1	0.1	0	0	0	0	0	0	0	0	0	0
9	0	0	0.1	0.1	0	0	0	0	0	0	0	0	0	0.4	0.4	0	0	0	0	0	0	0	0	0
10	0	0	0	0	0.6	0	0	0	0	0	0	0	0	0	0	0.3	0	0.1	0	0	0	0	0	0
11	0	0	0	0	0.2	0.1	0	0	0	0	0	0	0	0	0	0.5	0.1	0	0.1	0	0	0	0	0
12	0	0	0	0	0	0.1	0.5	0	0	0	0	0	0	0	0	0	0.2	0	0	0.2	0	0	0	0
13	0	0	0	0	0.1	0	0	0.1	0	0	0	0	0	0	0	0	0	0.7	0.1	0	0	0	0	0
14	0	0	0	0	0	0.1	0	0.1	0.4	0	0	0	0	0	0	0	0	0	0.3	0.1	0	0	0	0
15	0	0	0	0	0	0	0.3	0	0.4	0	0	0	0	0	0	0	0	0	0	0.3	0	0	0	0
16	0	0	0	0	0	0	0	0	0	0.3	0.5	0	0	0	0	0	0	0	0	0	0.1	0.1	0	0
17	0	0	0	0	0	0	0	0	0	0	0.1	0.2	0	0	0	0	0	0	0	0	0.5	0	0.2	0
18	0	0	0	0	0	0	0	0	0	0.1	0	0	0.7	0	0	0	0	0	0	0	0	0.1	0	0
19	0	0	0	0	0	0	0	0	0	0	0.1	0	0.1	0.3	0	0	0	0	0	0	0	0.3	0.2	0
20	0	0	0	0	0	0	0	0	0	0	0	0.2	0	0.1	0.3	0	0	0	0	0	0	0	0.4	0
21	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0.1	0.5	0	0	0	0	0	0	0.4
22	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0.1	0	0.1	0.3	0	0	0	0	0.5
23	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0.2	0	0.2	0.4	0	0	0	0.2
24	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0.4	0.5	0.2	0

Tableau 5.4 – Matrice de transitions représentative du treillis d'accès

5.5 Développement et validation

Pour assurer l'assistance à la navigation dans un cube OLAP, nous avons développé un prototype en Java, baptisé NAPARE (NAvigation PAth REcommendation). Notre méthode de navigation a été testée sur les données de l'entrepôt de données Foomart plus particulièrement le cube de données "Sales", et pour lequel une matrice représentant le treillis de cuboïdes a été construite.

NAPARE permet de simuler le déplacement d'un utilisateur dans le cube de données. Le principe est simple : on place l'utilisateur dans un nœud initial (correspond à sa première requête), puis on utilise les distributions de probabilités de transition

pour décider à chaque pas de temps où il se rend. L'idée est alors de garder le nombre de fois où l'utilisateur passe dans chaque nœud, ce qui permettrait de construire une distribution de probabilités (si convergence). Le principe de base est d'attribuer à chaque nœud une valeur (ou score) proportionnelle au nombre de fois que passerait par ce nœud un utilisateur parcourant le treillis de cuboïdes en cliquant aléatoirement, sur un des liens apparaissant sur chaque cuboïde. Ainsi, le déplacement de l'utilisateur est une marche aléatoire sur le graphe de cube de données. En sachant que l'utilisateur choisisse chaque nœud dépendant des nœuds précédemment visitées, il s'agit d'un processus de Markov.

Il est souvent intéressant, étant donné un modèle de Markov et une séquence d'observations $o = o_1, o_2, ...o_T$ de déterminer la séquence d'états $x = x_1, x_2, ..., x_T$ la plus probable ou le chemin le plus probable (ayant pu générer ces observations o). Il existe deux solutions, la première consiste déterminer toutes les séquences d'états ayant pu générer o, puis de calculer leur probabilités afin de déterminer la plus probable. Cependant, cette méthode est particulièrement coûteuse car, dans le cas général, il existe N^T chemins possibles. Ainsi, la deuxième solution s'impose. Elle consiste à utiliser le treillis (algorithme de Viterbi). Dans notre cas, nous utilisons le treillis de cuboïdes (sa matrice d'adjacence).

Nous nous sommes par ailleurs intéressés à l'évaluation de la performance de notre système. Tout d'abord, notre première expérience évalue la performance de l'approche proposée pour suggérer des recommandations. Notre analyse de performance consiste à évaluer le temps nécessaire pour générer une recommandation. Le temps est calculé pour différentes tailles de log. La taille de log consiste au nombre de sessions d'analyse total contenues dans ce log. Ainsi, la recommandation est calculée pour différentes tailles des logs. En effet, nous avons testé notre prototype sur une charge de 40 sessions d'analyse relatives à l'entrepôt de données *Foodmart* utilisé dans notre exemple. Ces sessions d'analyse comptent 120 requêtes. Lorsqu'une nouvelle session est effectuée, la charge doit être mise à jour. La Figure 5.9 montre que le temps nécessaire pour générer une recommandation est influencé par la taille des logs des sessions. Il augmente linéairement avec la taille des logs mais reste toujours acceptable puisqu'il ne dépasse pas la seconde pour le système NAPARE (au niveau de la navigation dans le cube de données).

La deuxième expérience que nous avons menée consiste à mesurer la précision de la recommandation. Notre système de recommandation repose sur les prédictions générées par le modèle Markov. Toutefois, ce dernier est un modèle statistique ce qui nous a amené à utiliser une mesure statistique de précision : MAE (Mean Absolute Error) [SM95] qui consiste à évaluer la différence existante entre les chemins prédits et les chemins réellement empruntés par les utilisateurs. La MAE calcule, pour chaque paire < chemin-prédiction >, la moyenne d'erreur absolue entre les chemins prédits $Pred(u_a, i)$ et les chemins réellement empruntés par les utilisateurs $v(u_a, i)$. n représente le nombre total de chemins prédits. Plus la valeur de MAE est faible, plus les prédictions sont précises et le système de recommandation est performant.

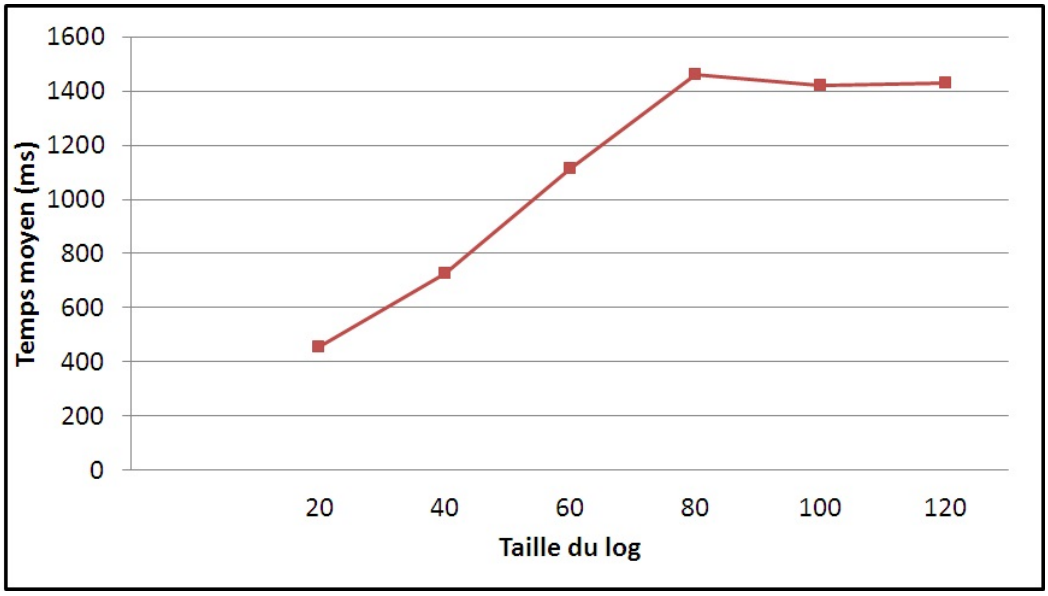


Figure 5.9 – Analyse de performance : Temps d'exécution de NAPARE en fonction de la taille du log

$$MAE = \frac{\sum_{i=1}^{n} |v(u_a, i) - Pred(u_a, i)|}{n}$$

Les résultats de NAPARE sont de bonne qualité puisque nous utilisons le treillis de cuboïdes qui nous garantit d'avoir toujours des chemins valides ou logiques. Il y aura plus de détail dans le Chapitre 6.

5.6 Conclusion

Avec le développement des entrepôts de données et l'analyse en ligne, nous nous intéressons dans ce chapitre à la prise en compte de l'utilisateur lors de sa navigation dans un cube de données. Ainsi, nous avons guidé l'utilisateur vers des chemins de navigation pertinents pour lui en se basant sur les précédentes sessions d'analyse de tous les utilisateurs de cube de données.

A notre connaissance, nous avons été les premiers à recommander des chemins de navigations au sein d'un cube de données OLAP. L'idée d'utiliser l'historique des autres utilisateurs pour suggérer des recommandations à l'utilisateur courant est très populaire dans le domaine de recherche d'information et de Web usage mining. Dans nos travaux précédents (Chapitre 4), nous avons aidé l'utilisateur à écrire sa requête initiale avec un système de recommandation qui repose sur l'historique des utilisateurs. Notre objectif est d'aider l'utilisateur son cube de données. Cependant, dans ce chapitre, nous intervenons pendant la phase de navigation au sein de cube OLAP déjà construit.

Nous avons montré dans ce chapitre que nous pouvons rapidement trouver les chemins qui peuvent apporter des réponses à une requête de navigation et surtout

de présenter à l'utilisateur les chemins pertinents parmi l'ensemble de tous les chemins en se basant sur un modèle probabiliste (Markov). Le calcul de chemin le plus intéressant se fait en temps réel au fur et à mesure de la navigation de l'utilisateur.

Nous avons développé un prototype baptisé NAPARE. Les tests que nous avons effectués sur NAPARE ont démontré que notre algorithme de recommandation de chemins de navigation donne de résultats satisfaisants. En effet, le temps de génération ne dépasse pas la seconde. Nous avons observé aussi que le temps d'exécution augmente quand la taille du log augmente. Nous avons également observé que les recommandations générées par NAPARE sont de bonne qualité tant qu'elles présentent des suggestions logiques.

Une perspective directe de ce travail est de procéder à l'élagage du treillis de cuboïdes, vu la taille de cube de données. En effet, le processus d'élagage consiste à éliminer les nœuds inintéressants du treillis d'accès (nœuds inexistants dans le log) que les utilisateurs ne visitent pas afin d'alléger leurs tâches de navigation.

Par ailleurs, nous voudrions suggérer des recommandations qui ne se basent pas seulement sur ce que les autres utilisateurs ont fait (l'historique des utilisateurs) mais aussi elles soient en adéquation avec le profil de l'utilisateur. C'est ce qu'on appelle recommandation personnalisée (Chapitre 2).

A court terme, nous voudrions aussi mener nos expériences sur des données réelles afin d'améliorer la qualité des recommandations réalisées.

Chapitre 6

Développement : réalisation et validation de nos contributions

Ce chapitre a pour but de présenter les développements informatiques que nous avons réalisés dans le but de valider nos contributions en terme de prise en compte de l'utilisateur durant ses analyses OLAP. Tout d'abord, nous avons été amenés à réaliser une plateforme logicielle permettant de prendre en compte l'utilisateur avant, pendant et après l'analyse OLAP. Ainsi, nous donnons les différents éléments concernant la conception et le développement de cette plateforme que nous avons appelée *PersoReco*. Ce chapitre est organisé comme suit. Dans un premier temps, nous décrivons l'architecture de notre système dans la section 6.1. Puis, nous présentons les expériences et leurs résultats dans la section 6.2.

6.1 Notre système : PersoReco

Notre système de prise en compte de l'utilisateur durant l'analyse OLAP est développé en Java sous JRE 1 avec SQL Server, Oracle et Jaspersoft. Tous les tests ont été réalisés avec un i5 4GB de RAM utilisant Windows 7 professionnel.

Notre plateforme *PersoReco* offre différentes fonctionnalités qui permettent la réalisation de notre approche de prise en compte de l'utilisateur durant ses analyses OLAP. En effet, notre système fonctionne de la manière suivante : à partir des préférences utilisateur il peut fournir un contenu adapté à l'utilisateur et d'autre part à partir de la requête courante et le log de requêtes, il peut générer des recommandations pendant l'écriture de la requête et aussi pendant la navigation.

La Figure 6.1 montre l'architecture globale de notre application et illustre ses différentes fonctionnalités pour l'utilisateur : (1) personnalisation du contenu baptisée *Perso* et (2) génération des recommandations baptisée *Reco*. La fonctionnalité de personnalisation de contenu appelée *Perso* présente la création de nouveau niveaux d'analyse par l'opérateur *PRoCK*. La fonctionnalité de recommandation se décline en deux sous fonctionnalités à savoir l'assistance à l'écriture de la requête avec le système *FIMIOQR* et la recommandation de chemins de navigation assurée par le système *NAPARE*.

PersoReco fonctionne de la manière suivante. Premièrement, l'utilisateur saisit

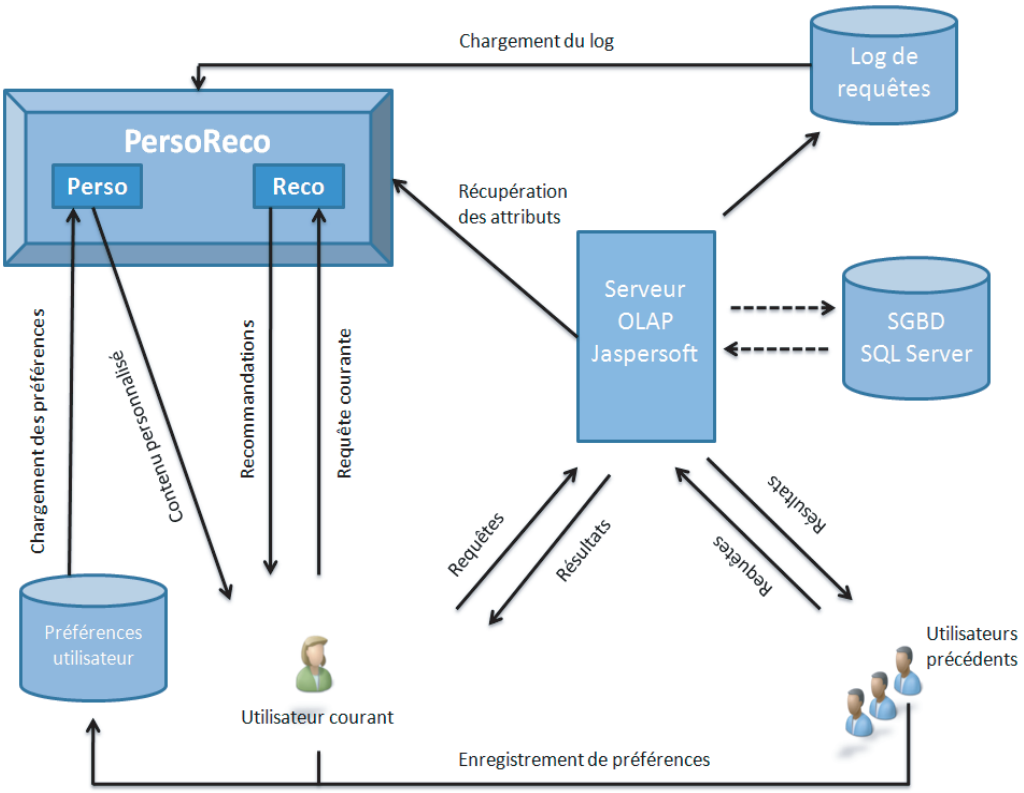


Figure 6.1 – Architecture du système PersoReco

ou met à jour ses préférences qui seront enregistrées dans la base de données de préférences. Ainsi l'application *PersoReco* et plus spécifiquement la fonctionnalité *Perso* adapte le contenu selon les préférences enregistrées et elle peut aller jusqu'à l'enrichir. Deuxièmement, l'utilisateur va écrire une requête pour la lancer sur l'entrepôt de données stocké dans le SGBD SQL Server. Les requêtes sont lancées par l'utilisateur via le serveur OLAP Jaspersoft. Ainsi, l'application *PersoReco* et plus particulièrement la fonctionnalité *Reco* lui assiste pendant l'écriture via les recommandations. Les requêtes précédentes que ce soit de l'utilisateur courant ou des autres utilisateurs sont enregistrées dans un log de requêtes qui est chargé par l'application *PersoReco*. Notre système va s'en servir pour générer les recommandations afin d'assister l'écriture en premier lieu et la navigation en deuxième lieu.

6.1.1 Fonctionnalité de personnalisation de contenu : *Perso*

La personnalisation de contenu réside dans la création de nouveaux niveaux de hiérarchie (*PRoCK*).
Nous avons implémenté l'opérateur *PRoCK* en PL/SQL au sein du Oracle 11g. Le résultat est un package baptisé *PRoCK* offrant toutes les fonctionnalités permettant de créer des hiérarchies de dimension personnalisées en tenant compte les préférences utilisateur. Ainsi, nous avons paramétré ce package pour qu'il s'adapte sur n'importe quel entrepôt de données. Nous avons conçu aussi une interface qui permet à un

utilisateur de saisir une dimension, des partitions et ses contraintes pour appliquer ensuite l'opérateur *PRoCK*.

Pour implémenter PRoCK au sein du SGBD Oracle, nous avons utilisé 3 tables relationnelles pour stocker les ensembles de données : PRoCK_partition, PRoCK_constraint et PRoCK_means (Chapitre 3). Après le processus de classification, la nouvelle hiérarchie est construite en utilisant des opérateurs SQL : le nouveau niveau est créé avec la commande *create table* et la fonction *roll-up* est établie avec une association clé primaire/clé étrangère entre le nouveau niveau et le niveau existant.

PRoCK est un package composé de 32 fonctions PL/SQL. Le package est implémenté sur Oracle. Pour cela, il faut nécessairement se connecter, tout d'abord, au compte Oracle. Il suffit de remplir l'ensemble des informations afin de mettre en place la connexion entre le compte Oracle et l'application Java. Une fois la connexion est effectuée, l'interface illustrée dans la Figure 6.2 se présente et nous permet de suivre les étapes d'exécution du package.

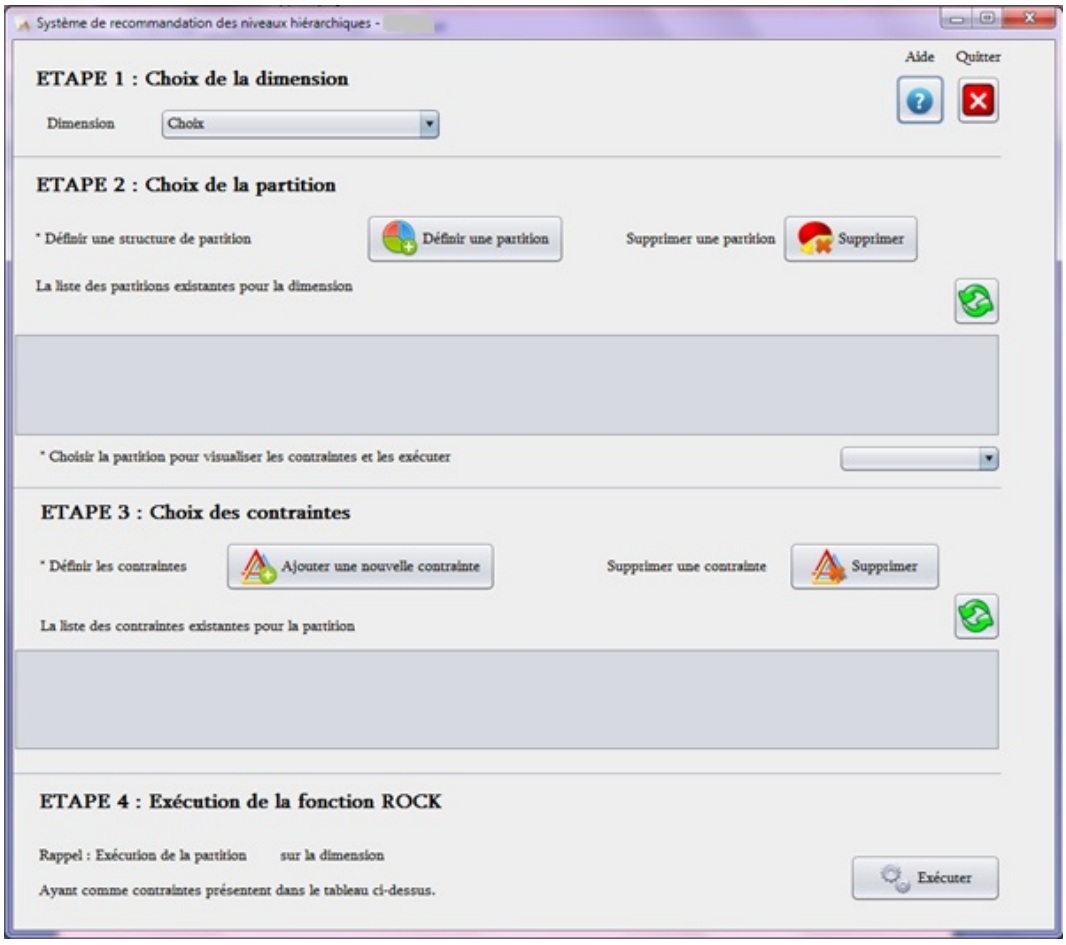


FIGURE 6.2 – PRoCK : Page de visualisation

Sur cette interface (Figure 6.2), il est possible de sélectionner une dimension afin de visualiser les partitions existantes sur cette dimension. Pour cela, il faut préciser la

dimension qu'on souhaite visualiser dans la liste des dimensions. Dans notre exemple, nous choisirons la table "position" (Figure 6.3).

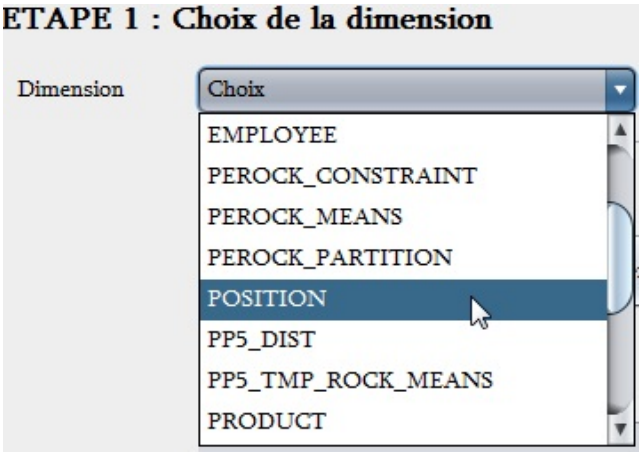


Figure 6.3 – PRoCK : Choix de la dimension

Ceci complète alors le tableau des dimensions (Figure 6.4) et le bouton vert (rafraîchir) nous permet de rafraichir l'affichage suite à la création ou la suppression d'une partition.

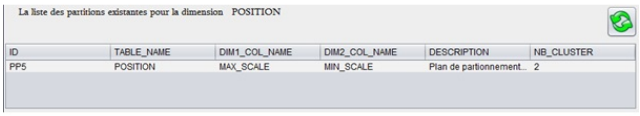


Figure 6.4 – PRoCK : Liste des partitions existantes

Si aucune structure de partition n'est présente, nous pouvons alors en créer une (Figure 6.5). Il faut donc cliquer sur le bouton ajouter une partition et remplir l'ensemble des champs suivants :

- libellé : nom de la partition (nom unique) ;
- description : description de la partition ;
- nom de la table : nom de la table de référence ;
- nombre de clusters : nombre de partitions ;
- nom de la première dimension : première dimension à analyser ;
- nom de la seconde dimension : deuxième dimension à analyser ;
- préfixe du nom des clusters : nom des partitions.

Une fois la partition est ajoutée, il faut rafraîchir le tableau de manière à la voir apparaître. Il est aussi possible de supprimer une partition.

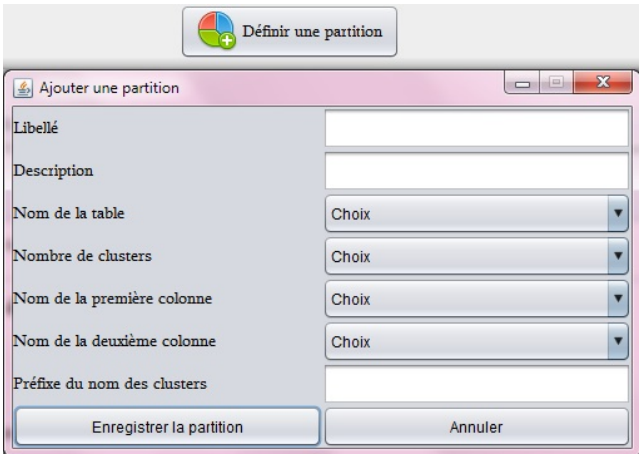


FIGURE 6.5 – PRoCK : Définition de la partition

Après avoir créé au moins une partition, il est possible de rajouter des contraintes sur les données (Figure 6.6). Il faut donc ajouter une contrainte et compléter les champs suivants :

- libellé : nom de la contrainte (nom unique) ;
- partition : référence à la partition créée ;
- élément 1 : identifiant dans la table d'un premier élément ;
- élément 2 : identifiant dans la table d'un second élément ;
- type de lien : deux types possible : "must link" c'est-à-dire les éléments doivent être ensemble et "cannot link" signifie que les éléments ne doivent pas être ensemble.

Une fois la contrainte est ajoutée, il faut rafraîchir le tableau de manière à la voir apparaître et il aussi possible de supprimer une contrainte.

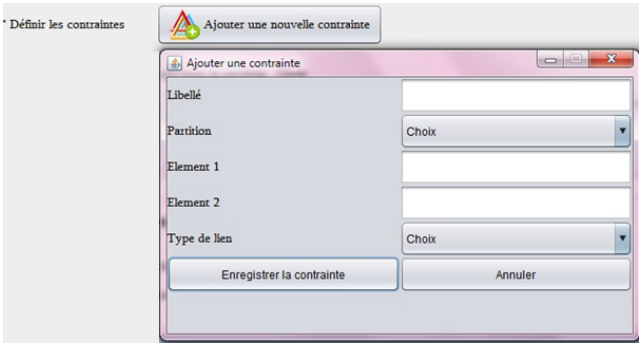


FIGURE 6.6 – PRoCK : Ajout d'une contrainte

Ainsi, nous pouvons visualiser les contraintes présentes sur une partition en la choisissant dans la liste (Figure 6.7).

Après avoir défini les structures de partitions et contraintes, nous pouvons alors exécuter l'application PRoCK en complétant le nom de la partition correspondante, ainsi que l'identifiant de la table correspondante. Nous pouvons même visualiser le résumé des éléments choisis avant d'exécuter PRoCK (Figure 6.8).

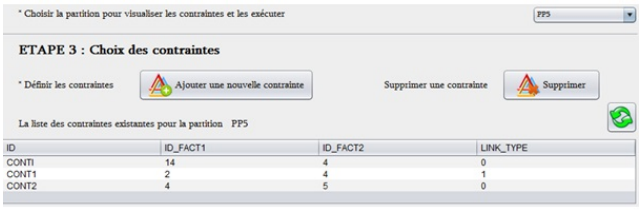


Figure 6.7 – PRoCK : Visualisation des contraintes

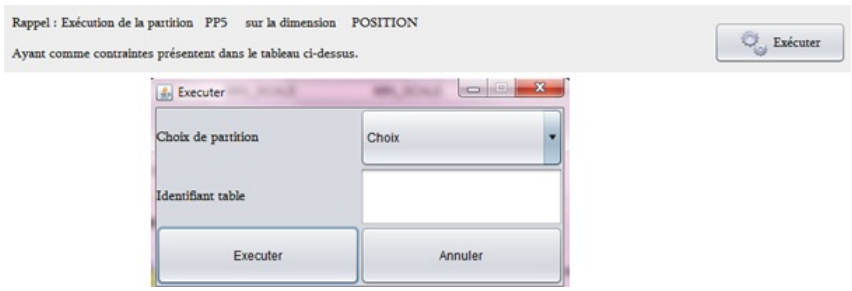


Figure 6.8 – PRoCK : Vérification des choix

Ainsi, l'application initialisera les centroïdes avant de calculer les distances et classer les données dans les différentes classes et nous pouvons donc visualiser les résultats obtenus par l'application (Figure 6.9).

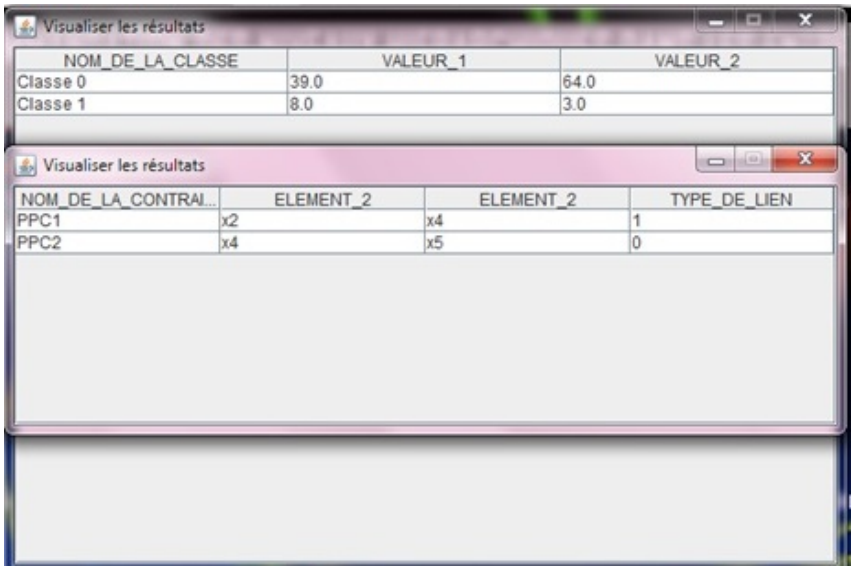


Figure 6.9 – PRoCK : Affichage des résultats

6.1.2 Fonctionnalité de recommandation : *Reco*

Assistance à l'écriture de requêtes

Pour assurer l'assistance à l'écriture de requêtes, nous avons développé un prototype en Java, baptisé *FIMIOQR* (Frequent Itemset Mining for Interactive OLAP Query recommendation).

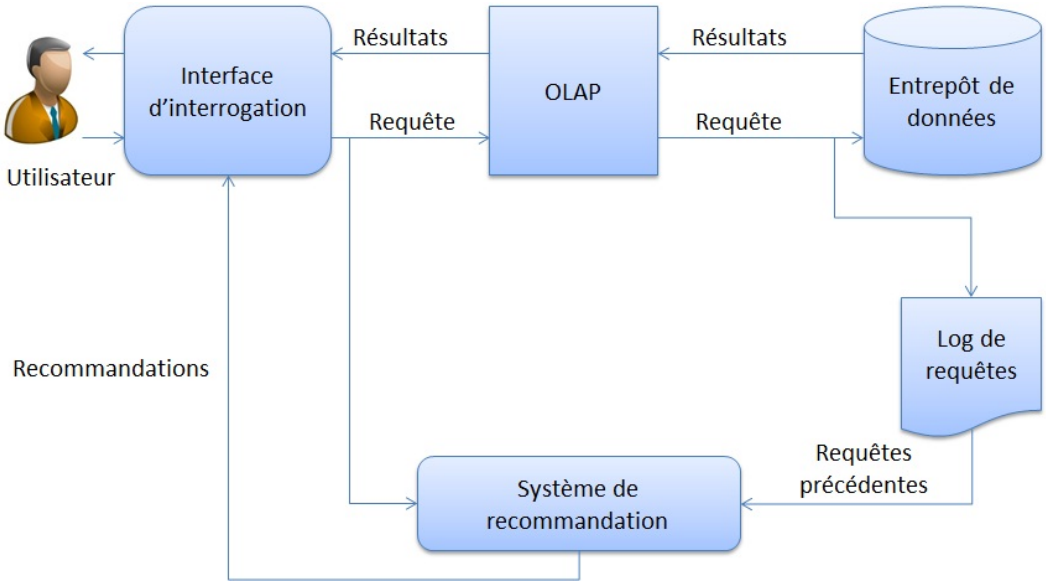


FIGURE 6.10 – Architecture de FIMIOQR

Notre système représente une couche logicielle intermédiaire (middleware-layer) au sommet du SGBD (SQL Server). Le fonctionnement de *FIMIOQR* est illustré dans la Figure 6.10. Les requêtes de l'utilisateur courant sont transférées via l'interface de l'entrepôt de données (Interface d'interrogation) à l'entrepôt de données et aussi à notre système. L'entrepôt de données traite la requête et retourne les résultats. En même temps la requête est enregistrée dans le log de requêtes. Ce log de requêtes est traité hors ligne pour en extraire les différents attributs des différentes requêtes. A chaque fois, l'utilisateur accède au système, le système de recommandation combine son input avec les motifs fréquents et génère un ensemble de recommandations.

Dans ce cadre, *FIMIOQR* permet l'écriture pas à pas de la requête. L'assistance à l'écriture se fait clause par clause comme illustré dans les Figures 6.11, 6.12 et 6.13.

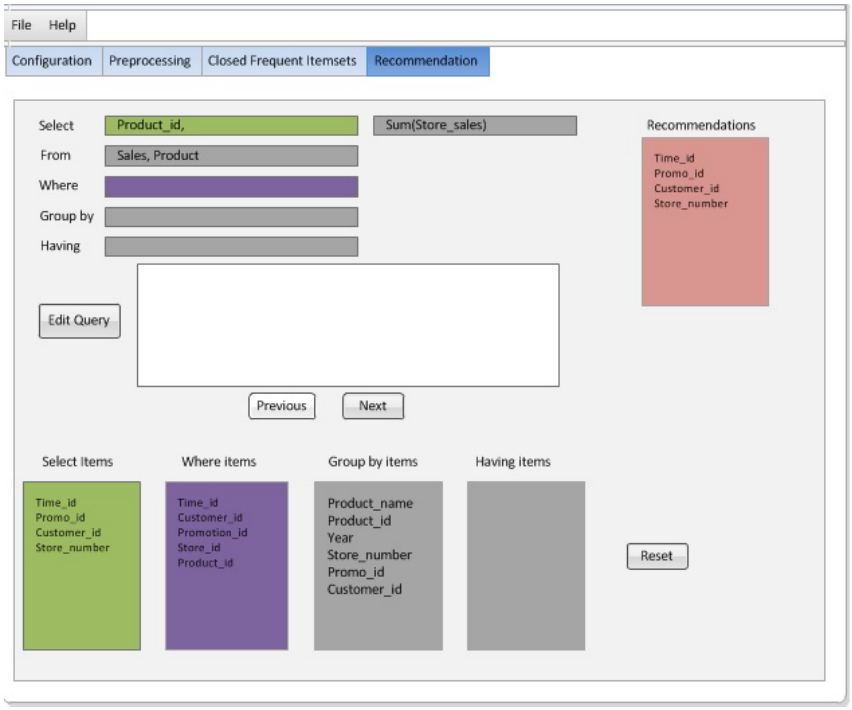


Figure 6.11 – FIMIOQR : clause Select

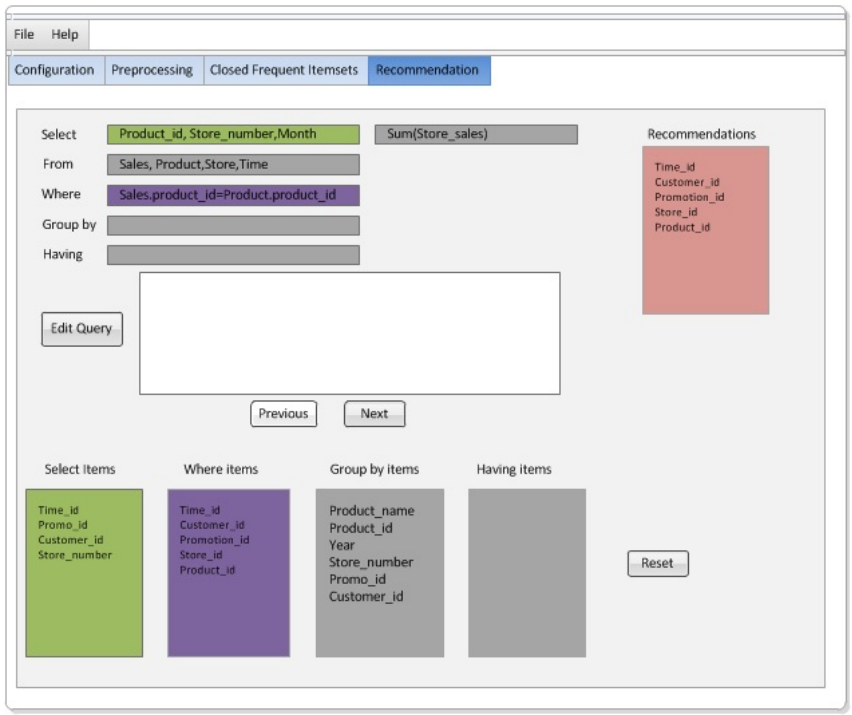


Figure 6.12 – FIMIOQR :clause Where

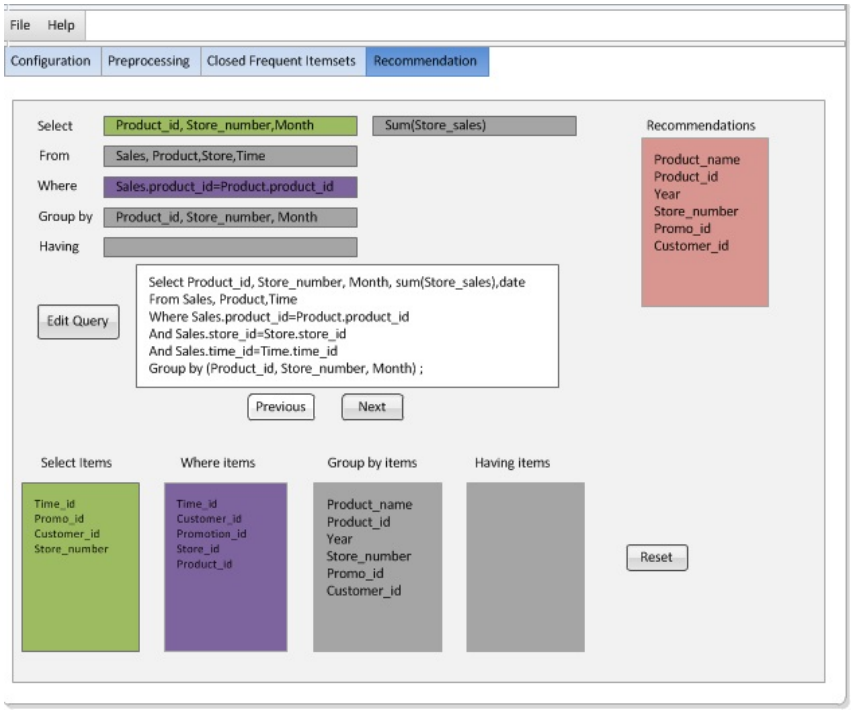


FIGURE 6.13 – FIMIOQR :clause Group by

Assistance à la navigation

Pour assurer l'assistance à la navigation dans un cube de données OLAP, nous avons développé un prototype en Java, baptisé *NAPARE* (NAvigation PAth REcommendation).

NAPARE permet de simuler le déplacement d'un utilisateur dans le cube de données. Le principe est simple : on place l'utilisateur dans un nœud initial (correspond à sa première requête), puis on utilise les distributions de probabilités de transition pour décider à chaque pas de temps où il se rend. L'idée est alors de garder le nombre de fois où l'utilisateur passe dans chaque nœud, ce qui permettrait de construire une distribution de probabilités (si convergence). Le principe de base est d'attribuer à chaque nœud une valeur (ou score) proportionnelle au nombre de fois que passerait par ce nœud un utilisateur parcourant le treillis de cuboïdes en cliquant aléatoirement, sur un des liens apparaissant sur chaque cuboïde. Ainsi, le déplacement de l'utilisateur est une marche aléatoire sur le graphe de cube de données. En sachant que l'utilisateur choisisse chaque nœud dépendant des nœuds précédemment visitées, il s'agit d'un processus de Markov.

Nous nous sommes par ailleurs intéressés à l'évaluation de la performance de notre système *PersoReco*.

6.2 Expériences et résultats

Nous présentons dans cette section les expériences menées sur les deux grandes fonctionnalités de notre plateforme de prise en charge de l'utilisateur à savoir la personnalisation de contenu *Perso* et l'assistance de l'utilisateur via les recommandations *Reco*.

6.2.1 Les données

Notons que notre application *PersoReco* a besoin de contraintes utilisateur pour la fonctionnalité de personnalisation et du log de requêtes pour générer des recommandations. Par conséquent, nous utilisons la collecte explicite des contraintes utilisateurs, c'est-à-dire c'est à l'utilisateur de saisir ses contraintes et de les mettre à jour. Par contre, pour la fonctionnalité de recommandation, notre système utilise la collecte implicite des données c'est-à-dite c'est à notre système de charger le log de requêtes et y fouiller pour générer les recommandations à l'utilisateur.

Pour cela, nous utilisons le log de requêtes de l'entrepôt de données *FoodMart* fourni avec le moteur OLAP JasperSoft SQL Management.

6.2.2 Analyse de performance de PRoCK

Pour évaluer notre système *PersoReco*, en premier temps, nous avons analysé la performance de l'opérateur *PRocK* par la mesure du temps nécessaire pour générer un nouveau niveau d'analyse. La performance d'un système de personnalisation peut être évaluée en terme de temps de calcul. Il s'agit d'un temps de calcul réel qui permet d'évaluer le temps requis pour l'exécution des algorithmes et l'obtention des résultats escomptés.

Il va de soi que la mesure du temps de calcul est dépendante des spécifications matérielles de la machine utilisée pour l'exécution de ces calculs, ainsi que des programmes et applications lancés simultanément sur cette machine au moment des calculs. En ce qui concerne nos expériences, elles ont été réalisées sur un PC avec Windows 7 professionnel, ayant 4 Go de RAM et un processeur i5. En effet, nous avons testé *PRocK* sur l'entrepôt de données *Foodmart*. *PRocK* nécessite juste quelques minutes pour générer un nouveau niveau de hiérarchie et le matérialiser.

6.2.3 Analyse de performance de système de recommandation

La performance d'un système de recommandation peut être également évaluée en terme de temps de calcul. Il s'agit d'un temps de calcul réel qui permet d'évaluer le temps requis pour l'exécution des algorithmes et l'obtention des résultats escomptés. Il va de soi que la mesure du temps de calcul est dépendante des spécifications matérielles de la machine utilisée pour l'exécution de ces calculs, ainsi que des programmes et applications lancés simultanément sur cette machine au moment des calculs. En ce qui nous concerne, notre analyse de performance consiste à évaluer le temps nécessaire pour générer une recommandation. Le temps est calculé pour différentes tailles de log. La taille de log consiste au nombre total des requêtes contenues dans ce log. Ainsi, la recommandation est calculée pour différentes tailles des logs.

En effet, nous avons calculé le temps nécessaire de génération d'une recommandation pour différentes tailles de logs relatifs à l'entrepôt de données *Foodmart* utilisé dans notre exemple au niveau de l'assistance de formulation de requêtes et au niveau de l'assistance à la navigation réalisées par les deux systèmes *FIMIOQR* et *NAPARE*.

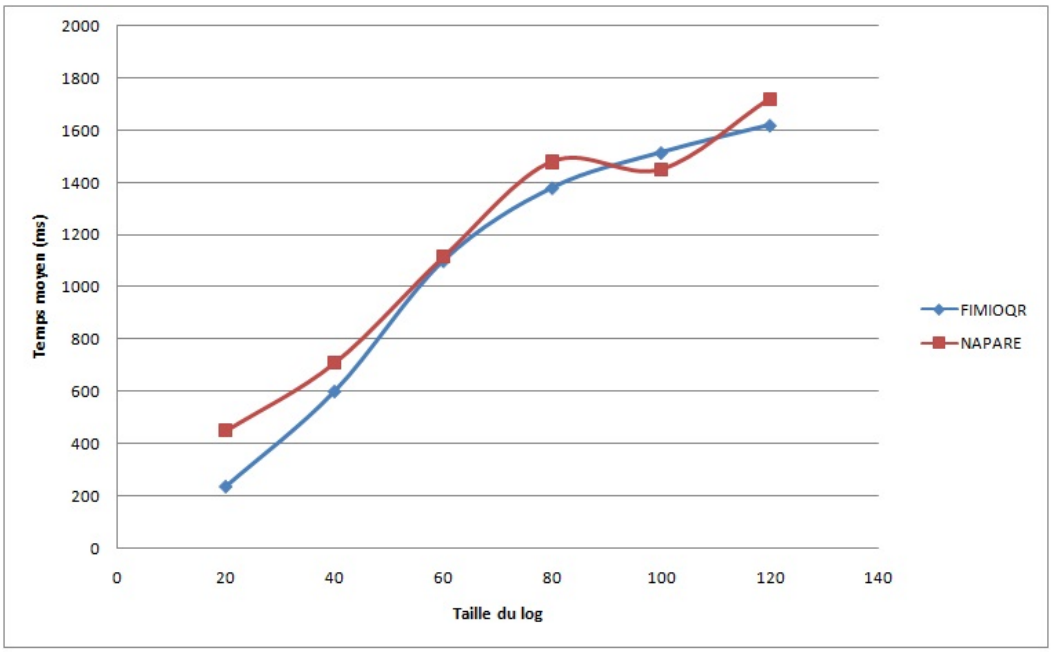


FIGURE 6.14 – Analyse de performance : Temps d'exécution en fonction de la taille du log

La Figure 6.14 montre que le temps nécessaire pour générer une recommandation est influencé par la taille des logs des requêtes. Il augmente linéairement avec la taille des logs mais reste toujours acceptable puisqu'il ne dépasse pas la seconde pour les deux systèmes *FIMIOQR* et *NAPARE*.

6.2.4 Évaluation de la qualité de la recommandation

L'évaluation des systèmes de recommandation constitue une étape clé dans un processus de recommandation dans la mesure où elle reflète la performance de l'intégralité du système. Pour tout système de recommandation, prédire efficacement les futures appréciations contribue à la satisfaction des besoins des utilisateurs. Pour déterminer l'efficacité d'un système de recommandation, des indicateurs comme la précision, le rappel, le MAE, le HMAE et la couverture sont utilisés. Le choix de telle ou telle métrique dépend notamment de la problématique de départ, des objectifs escomptés et de la nature de l'expérience à mener.

6.2.4.1 Précision des recommandations

La plupart des travaux de recherche portant sur les systèmes de recommandation, évaluent la performance de leurs algorithmes en s'appuyant notamment sur le

critère de précision des prédictions. En effet, un système de recommandation peut recommander des éléments intéressants ou non intéressants. La précision permet alors d'évaluer la capacité du système à recommander des éléments que l'utilisateur apprécie réellement. Il est à signaler que la qualité et la précision des recommandations sont étroitement liées à la disponibilité des données sur les appréciations. En effet, quand ces données sont rares, le système ne peut pas générer des prédictions précises. La mesure de précision représente à quel point les instances recommandées par une méthode sont réellement intéressantes pour les utilisateurs. En bref, elle indique si les éléments recommandés sont vraiment intéressants c-à-d cette mesure évalue si un élément sélectionné par un utilisateur est réellement perçu comme étant pertinent par ce même utilisateur. La précision est un indicateur qui représente la qualité de la recommandation, c'est-à-dire à quel point les suggestions proposées sont conformes aux intérêts de l'utilisateur. Ainsi, l'indicateur de précision permet de déterminer la probabilité qu'un élément recommandé soit pertinent. La précision générale du système de recommandation correspond ainsi à la moyenne des précisions calculées pour chaque utilisateur actif. Plus cette précision est élevée, plus le système de recommandation est performant.

$$Précision = \frac{nbre\ des\ suggestions\ pertinentes}{nbre\ des\ suggestions}$$

6.2.4.2 Rappel

Le rappel indique combien de suggestions pertinentes ont été recommandées à l'utilisateur par rapport au nombre total de suggestions pertinentes disponibles dans le système. Il permet de déterminer la probabilité qu'un élément pertinent soit recommandé.

$$Rappel = \frac{nbre\ des\ suggestions\ proposées\ à\ l'utilisateur}{nbre\ des\ suggestions\ pertinentes}$$

On utilise aussi des techniques statistiques pour mesurer l'efficacité d'un algorithme de recommandation. L'idée c'est d'évaluer la précision de la prédiction effectuée par le système en comparant les prédictions avec les choix qu'aurait fourni l'utilisateur dans un cas réel.

6.2.4.3 MAE

Pour évaluer la qualité de la prédiction, MAE (Mean Absolute Error) ou l'Erreur Moyenne Absolue [SM95] est une mesure statistique de précision qui consiste à mesurer la déviation des recommandation prédites avec les choix réels effectués par les utilisateurs. Plus l'erreur moyenne absolue est faible, plus les prédictions sont précises et le système de recommandation est performant. La MAE calcule, pour chaque paire <élément-prédiction>, la moyenne d'erreur absolue entre les éléments prédits $Pred(u_a, i)$ et les éléments réellement choisis par les utilisateurs $v(u_a, i)$. Sachant que n représente le nombre total des éléments prédits, la MAE est définie comme suit :

$$MAE = \frac{\sum_{i=1}^{n} |v(u_a, i) - Pred(u_a, i)|}{n}$$

Nous utilisons cette métrique pour évaluer la prédiction de chemins de navigation. Les résultats sont de bonne qualité puisque nous utilisons le treillis de cuboïdes qui nous garantit de recommander des chemins valides ou logiques.

6.2.4.4 HMAE

Les systèmes de recommandations ont pour objectif de calculer les prédictions des notes manquantes concernant le maximum de paires <utilisateur-élément>. Une fois ces prédictions calculées, les éléments ne sont pas tous recommandés par la suite aux utilisateurs. En effet, seuls les éléments ayant les valeurs de prédiction les plus élevées sont proposés. Dans ce cas, l'erreur concernant les éléments ayant de faibles valeurs de prédiction n'est pas utile quant à l'évaluation de la performance des systèmes de recommandation, tandis que l'erreur relative aux éléments ayant des notes prédites élevées est d'une grande importance

6.2.4.5 La couverture

La couverture mesure la capacité du système à fournir des recommandations. Pour les systèmes de recommandation basés sur le contenu, la couverture peut être évaluée par rapport à la capacité du système de recommandation à générer des prédictions pour toutes les scores manquants au niveau de la matrice de scores "Utilisateur x Élément". Elle peut être également évaluée en prenant en considération uniquement les prédictions contenues dans le corpus test. En effet, dans certains cas, le système de recommandation exploitant le filtrage collaboratif, peut être incapable de calculer les recommandations. Cette incapacité peut notamment être engendrée par le manque de données. En effet, faute de scores provenant des autres utilisateurs, le système aura des difficultés à calculer certaines prédictions. Ainsi, un système de recommandation ne peut être performant que lorsqu'il est susceptible de calculer un nombre suffisant de prédictions concernant un maximum d'utilisateurs. Autrement dit, le système doit pouvoir répondre aux attentes des différents utilisateurs actifs présents dans le système.

La Figure 4.11 montre que la précision augmente légèrement avec la taille du log de requêtes. Cette figure démontre que les la majorité des requêtes courantes peuvent obtenir des recommandations réussies dans notre approche avec une précision comprise entre 0.18 et 0.68.

Nous observons également que les recommandations générées que ce soit par FIMIOQR ou NAPARE sont de très bonne qualité. Nos tests montrent que nous pouvons générer des requêtes et des chemins de navigations pertinents en un temps d'exécution convenable.

Pour conclure, la meilleure mesure de l'efficacité d'un algorithme de recommandation et de la pertinence des suggestions c'est finalement la satisfaction de l'utilisateur, qui n'est pas toujours facile à bien identifier.

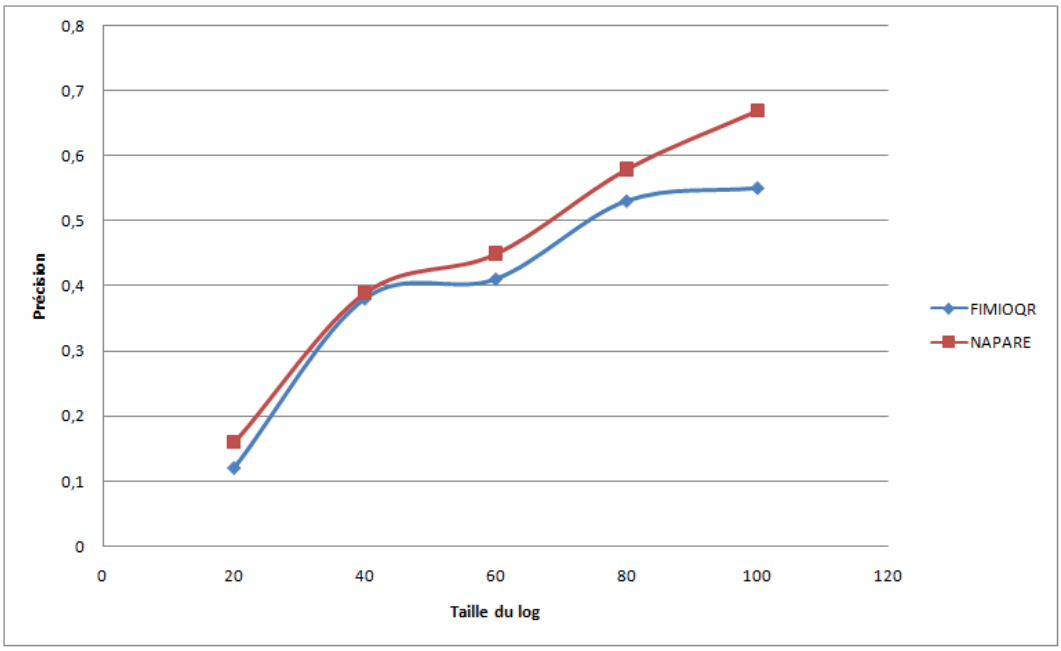


Figure 6.15 – Précision de recommandation en fonction de la taille du log

6.3 Conclusion

Nous avons présenté dans ce chapitre le système développé ainsi que son architecture. Ce système nous a permis de valider notre approche de prise en compte de l'utilisateur dans les entrepôts de données. Nous avons expérimenté notre système de prise en compte de l'utilisateur à différents moments de l'analyse OLAP et les tests que nous avons effectués ont démontré que les deux fonctionnalités de notre système à savoir la personnalisation et la recommandation donnent de résultats satisfaisants. La fonctionnalité de personnalisation permet de créer un contenu adapté à l'utilisateur plus particulièrement à ses contraintes. La fonctionnalité de recommandation permet d'assister l'utilisateur pendant son analyse OLAP et lui recommander des chemins de navigation pertinents en un temps d'exécution très convenable. En effet, le temps de génération ne dépasse pas la seconde. Nous avons observé aussi que le temps d'exécution augmente quand la taille du log augmente. Nous avons également observé que que les recommandations générées que ce soit par FIMIOQR ou NAPARE sont de très bonne qualité.

Chapitre 7

Conclusion générale

Dans ce chapitre consacré à la conclusion générale, nous faisons tout d'abord un bilan de nos travaux de recherche et indiquons ensuite dans quelles directions ces travaux pourront être poursuivis.

7.1 Synthèse des travaux

La réussite du processus d'entreposage repose entre autres sur la prise en compte de l'utilisateur, puisque c'est ce dernier qui va réaliser l'analyse en ligne. Au cours de ce mémoire, nous avons considéré la personnalisation et la recommandation sous l'angle collaboratif. En ce sens, nous avons exploité les usages d'un utilisateur donné, pour que les autres utilisateurs appartenant à la même communauté puissent en tirer profit. En d'autres termes, il s'agit de donner la possibilité aux utilisateurs d'un système décisionnel de partager des analyses OLAP et ainsi de faire évoluer l'OLAP vers d'autres possibilités d'analyse plus proches des besoins réels de l'utilisateur en intégrant l'aspect collaboratif.

Dans ce mémoire, nous avons présenté trois contributions. Chacune d'entre elles est liée à un événement particulier du processus d'analyse en ligne. Tout d'abord, nous avons présenté une approche de personnalisation qui permet de prendre en compte les contraintes utilisateur et de les exploiter afin de restituer un contenu personnalisé. C'est la personnalisation dirigée par les contraintes utilisateurs (Chapitre 3). Nous pensons que les hiérarchies de dimension ont un effet important sur l'analyse OLAP. C'est pourquoi, elles doivent être prises en compte pour la personnalisation de l'OLAP. L'originalité de nos travaux consiste alors à personnaliser les hiérarchies de dimensions qui constitueront de nouveaux axes d'observation sémantiquement plus riches pour l'utilisateur. Par conséquent, nous pouvons fournir à l'utilisateur un contenu personnalisé pouvant lui permettre de procéder à de nouvelles explorations. Pour cela, nous avons conçu et développé un opérateur personnalisé appelé PRoCK (Personalized Roll-up with Constrained Kmeans). PRoCK propose de bénéficier de techniques de fouille de données, plus précisément du clustering contraint, pour découvrir de nouveaux regroupements de données tout en respectant les contraintes

utilisateur. Lorsqu'un nouveau niveau d'analyse est créé sur une table de dimension, les autres utilisateurs appartenant à la même communauté d'utilisateurs peuvent en tirer profit, dans l'esprit d'un système collaboratif dans lequel chacun apporte sa pierre à l'édifice.

Par conséquent, PRoCK nous permet d'impliquer davantage l'utilisateur avant qu'il entame son analyse OLAP. Cependant, pour avoir un système OLAP complètement centré utilisateur, il faut personnaliser l'analyse OLAP elle-même. Durant l'analyse OLAP, nous avons deux étapes primordiales, à savoir la formulation de la première requête décisionnelle pour construire un cube OLAP et la navigation dans le cube de données en utilisant les opérateurs OLAP (Roll-up, Drill-down, etc.). De ce fait, nous avons étendu le concept de personnalisation à la recommandation dans ces deux étapes cruciales de l'analyse OLAP.

En premier lieu, nous avons proposé une approche collaborative de formulation de requêtes décisionnelles pertinentes. Pour cela, nous avons utilisé la recommandation interactive de requêtes décisionnelles au fur et à mesure de l'écriture de la requête. Cette approche s'appuie sur la charge de requêtes associée à un ou un ensemble d'utilisateurs appartenant à la même communauté d'acteurs d'une organisation. Ainsi, nous avons aidé l'utilisateur à soumettre à l'entrepôt de données des requêtes pertinentes construites, pas à pas, à partir des attributs les plus fréquemment utilisés par l'ensemble des acteurs de la communauté d'utilisateurs à laquelle il appartient. Nous avons développé pour cela, un système de recommandation collaborative de requêtes décisionnelles pertinentes, FIMIOQR (Frequent Itemsets Mining for Interactive OLAP Query Recommendation) (Chapitre 4). L'originalité de ce système réside, d'une part, dans son aspect interactif et collaboratif et, d'autre part, dans l'utilisation de la fouille de données, et plus précisément la méthode d'extraction des motifs fréquents *Close* pour extraire les attributs fréquents utilisés par l'ensemble des utilisateurs du système décisionnel.

En deuxième lieu, toujours pour guider l'utilisateur dans une autre étape cruciale de l'analyse OLAP, à savoir la navigation, nous avons proposé une approche de recommandation de chemins de navigation. Cette approche s'appuie sur le modèle de Markov qui permet de prédire l'état suivant avec la seule connaissance de l'état présent. Ainsi, à partir d'un point de navigation (nœud courant dans le treillis de cuboïdes), et en utilisant la propriété des chaînes de Markov, nous avons prédit le point de navigation suivant à l'utilisateur. Ce travail s'inscrit dans une approche de recommandation collaborative car les probabilités de passage d'une requête à l'autre dans le treillis de cuboïdes a été calculé en tenant compte des toutes les requêtes d'analyse de l'ensemble des utilisateurs appartenant à la même communauté. Nous avons conçu pour cela un système de recommandation de chemins de navigation nommé NAPARE (NAvigation PAth REcommendation) qui s'appuie sur le modèle de Markov construit à partir de l'historique des sessions des analyses de l'ensemble des utilisateurs exploitant le même cube OLAP.

Nos travaux peuvent se résumer de la manière suivante : à partir des contraintes utilisateur, nous pouvons fournir un contenu adapté à l'utilisateur et, d'autre part, à partir de la requête courante et du log de requêtes de l'ensemble des utilisateurs appartenant à la même communauté, nous pouvons générer des recommandations d'une part pendant l'écriture de la requête décisionnelle pour créer le cube de données à analyser, et d'autre part pendant la navigation dans le cube de données déjà créé. Ainsi, nous pouvons résumer les résultats obtenus par nos travaux en trois points :

1. personnalisation du contenu assuré par l'opérateur PRoCK ;
2. assistance à l'écriture de requêtes décisionnelles assuré par le système FIMIOQR ;
3. assistance à la navigation dans le cube de données assuré par le système NAPARE.

En se référant au comparatif présenté dans le Chapitre 2 (Tableau 2.2), nous présentons nos trois contributions dans le Tableau 7.1.

		PRoCK	FIMIOQR	NAPARE
Orientation	Personnalisation	×		
	Recommandation		×	×
Type	Cognitif			
	Collaboratif	×	×	×
Profil	Préférences	×		
	Usage		×	×
	Contexte			
Moment (% interrogation)	Avant	×		
	Pendant		×	
	Après			×
Objet d'étude	Schéma	×		
	Requête		×	
	Interface		×	
Collecte d'information	Explicite	×		
	Implicite		×	×
Formulation de préférences	Quantitative			
	Qualitative	×		
Technique utilisée		Fouille de données : Clustering contraint	Fouille de données : Recherche des motifs fréquents	Chaînes de Markov

Tableau 7.1 – Synthèse de nos travaux

Nous avons expérimenté notre système de prise en compte de l'utilisateur à différents moments de l'analyse OLAP et les tests que nous avons effectués ont démontré que les deux fonctionnalités de notre système, à savoir la personnalisation et la recommandation, donnent des résultats satisfaisants. La fonctionnalité de personnalisation permet de créer un contenu adapté à l'utilisateur, plus particulièrement à ses contraintes, via l'opérateur PRoCK. La fonctionnalité de recommandation permet d'assister l'utilisateur pendant son analyse OLAP et de lui recommander des chemins de navigation pertinents en un temps d'exécution très convenable. En effet, le temps de génération ne dépasse pas la seconde. Nous avons aussi observé que le temps d'exécution augmente quand la taille du log augmente. Nous avons également observé que les recommandations générées que ce soit par FIMIOQR ou NAPARE, sont de très bonne qualité. Nos tests montrent que nous pouvons générer des requêtes et des chemins de navigations pertinents en un temps d'exécution convenable.

Nos travaux présentés dans ce mémoire constituent une base prometteuse d'assistance de l'utilisateur durant son analyse OLAP en étant au plus près de ses attentes. Cependant, certaines perspectives s'offrent à nous.

7.2 Perspectives de recherche

Dans la continuité directe de notre travail de thèse, nous pouvons dégager des orientations de recherche qui permettraient de consolider notre prise en compte de l'utilisateur. En particulier, nos efforts portent actuellement sur l'amélioration de la plateforme *PersoReco* qui contient tous les systèmes déjà développés, à savoir PRoCK, FIMIOQR et NAPAPRE.
À plus long terme, nous pourrions donner des versions plus générales de nos travaux de prise en compte de l'utilisateur pendant l'analyse OLAP en explorant les pistes suivantes :

- Application de nos approches sur des données réelles
- Prise en compte du contexte utilisateur
- Vers un filtrage collaboratif personnalisé
- Évaluation de la qualité de recommandation
- Entrepôt de données personnalisé dans le cloud

7.2.1 Application de nos approches sur des données réelles

Nos expériences ont été réalisées sur des données synthétiques. Par contre, afin d'améliorer la qualité des recommandations et surtout l'assistance de l'utilisateur pendant son analyse OLAP, il faudrait pouvoir mener les expériences sur des données réelles pour connaître les choix de l'utilisateur et le satisfaire au mieux.

7.2.2 Prise en compte du contexte utilisateur

Dans le cadre de nos travaux, nous avons utilisé le concept d'évolution de schéma pour adapter l'entrepôt de données aux nouveaux besoins de l'utilisateur. Ainsi, nous avons remarqué que la plupart des résultats d'analyse obtenus sont liés au contexte d'utilisation. De ce fait, nos efforts portent actuellement sur la prise en compte du contexte utilisateur. Tout d'abord, nous essayons de définir le contexte utilisateur pour le bien le représenter. Ensuite, le contexte peut être intégré soit dans le profil de l'utilisateur, soit dans le modèle même de l'entrepôt, sous forme de dimensions contextuelles. Enfin, le contexte utilisateur peut être exploité dans la phase d'interprétation des résultats.

7.2.3 Vers un filtrage collaboratif personnalisé

Dans nos travaux, nous avons considéré la recommandation sous l'angle collaboratif en exploitant les usages de l'ensemble des utilisateurs de la même communauté. Cependant, les système de recommandation du e-commerce Le filtrage collaboratif est une solution profitable pour les plateformes de e-commerce, qui maximisent leurs ventes. s A l'instar des plateformes de e-commerce qui utilisent le filtrage collaboratif personnalisé comme une solution pour maximiser leurs ventes, nous voudrions suggérer des recommandations qui ne se basent pas seulement sur ce que les autres utilisateurs ont fait (l'historique des utilisateurs) mais aussi elles soient en adéquation avec le profil de l'utilisateur. C'est ce qu'on appelle recommandation personnalisée (Chapitre 2).

7.2.4 Évaluation de la qualité de recommandation

Dans le contexte de tous les systèmes d'information (entreprises, commerce électronique, loisirs, etc.), la pertinence de l'information délivrée et son adaptation aux usages et préférences des clients constituent des facteurs clés du succès ou du rejet de ces systèmes. C'est dans ce cadre que nous envisageons d'engager nos recherches à venir pour améliorer le processus de recommandation dans les systèmes décisionnels grâce à l'évaluation qualitative de la recommandation. En effet, l'un des facteurs clé de la recommandation est la qualité des informations délivrées.

Concernant la qualité des données délivrées, il s'agit d'évaluer la qualité d'une analyse proposée en réponse à un besoin, notamment la proximité des résultats calculés avec les résultats attendus. Dans ce cas, il faut définir des critères de qualité adaptés. Ces derniers peuvent être inspirés des travaux menés en qualité des données et qualité des processus mais peuvent être définis également par les utilisateurs du système eux-mêmes.

Nous pensons que des avancées devraient être faites au niveau de l'expérimentation, surtout au niveau des métriques utilisées pour mesurer la qualité de recommandation. Généralement, deux métriques, précision et rappel, sont souvent utilisées pour évaluer la qualité de la recommandation [HK00].

Une autre idée consiste à faire une veille sur les recommandations les plus acceptées par l'utilisateur afin de pouvoir les classer et les recommander par la suite en priorité à l'utilisateur dans ses sessions futures. Cette option est motivée par

les observations faites durant les développements et les expériences : le nombre de recommandations est assez important.

7.2.5 Entrepôt de données personnalisé dans le cloud

Avec l'avènement des big data et des nouveaux environnements informatiques tels que le cloud computing, l'évolution des entrepôts et de l'analyse en ligne des données ne peut se réaliser, à notre avis, que par la cohabitation avec ces nouveaux environnements.

L'entrepôt de données reste un fondement de la Business intelligence des entreprises. Il permet de travailler la donnée dans le temps, mais il peut s'avérer beaucoup moins adapté lorsqu'il s'agit de travailler sur des données non structurées, dont le schéma est établi a posteriori. Face à cet enjeu, notre idée est de relâcher le modèle de l'entrepôt de données et de proposer une analyse OLAP plus élaborée facilitant ainsi le processus de personnalisation. Notre objectif est de poursuivre nos travaux dans cette voie et de tester notre approche de l'OLAP collaboratif dans le cadre des entrepôts et de l'analyse en ligne de données massives.

Bibliographie

[ADB+99] Gregory D. Abowd, Anind K. Dey, Peter J. Brown, Nigel Davies, Mark Smith, and Pete Steggles. Towards a better understanding of context and context-awareness. In *HUC*, pages 304–307, Karlsruhe, Germany, 1999.

[AGG+15] Julien Aligon, Enrico Gallinucci, Matteo Golfarelli, Patrick Marcel, and Stefano Rizzi. A collaborative filtering approach for recommending olap sessions. *Journal of Decision Support Systems*, Volume 69(0) :pages 20–30, 2015.

[AGPS02] Liliana Ardissono, Anna Goy, Giovanna Petrone, and Marino Segnan. Personalization in business-to-customer interaction. *Commun. ACM*, 45(5) :52–53, 2002.

[AMS+96] Rakesh Agrawal, Heikki Mannila, Ramakrishnan Srikant, Hannu Toivonen, and A. Inkeri Verkamo. Fast discovery of association rules. In *Advances in Knowledge Discovery and Data Mining*, pages 307–328. AAAI/MIT Press, 1996.

[ASS01] Alberto Abelló, José Samos, and Fèlix Saltor. A framework for the classification and description of multidimensional data models. In *DEXA*, pages 668–677, Munich, Germany, 2001.

[AT05] Gediminas Adomavicius and Alexander Tuzhilin. Toward the next generation of recommender systems : A survey of the state-of-the-art and possible extensions. *Journal of IEEE Trans. Knowl. Data Eng.*, Volume 17(6) :pages 734–749, 2005.

[BF09] Fadila Bentayeb and Cécile Favre. Rok : Roll-up with the k-means clustering method for recommending olap queries. In *DEXA*, pages 501–515, Linz, Austria, 2009.

[BFB08] Fadila Bentayeb, Cécile Favre, and Omar Boussaid. A user-driven data warehouse evolution approach for concurrent personalized analysis needs. *Integrated Computer-Aided Engineering*, 15(1) :21–36, 2008.

[BGM+05] Ladjel Bellatreche, Arnaud Giacometti, Patrick Marcel, Hassina Mouloudi, and Dominique Laurent. A personalization framework for olap queries. In *DOLAP*, pages 9–18, Bremen, Germany, 2005.

[BHMD05] Ricardo A. Baeza-Yates, Carlos A. Hurtado, Marcelo Mendoza, and Georges Dupret. Modeling user search behavior. In *LA-Web*, pages 242–251, Buenos Aires, Argentina, 2005.

[BK05] Mokrane Bouzeghoub and Dimitre Kostadinov. Personnalisation de l'information : aperçu de l'état de l'art et définition d'un modèle flexible de profils. In *CORIA*, pages 201–218, Grenoble, France, 2005.

[BK13] Fadila Bentayeb and Rym Khemiri. Adapting OLAP analysis to user's constraints through semantic hierarchies. In *ICEIS*, pages 193–200, Angers, France, 2013.

[BKS01] Stephan Börzsönyi, Donald Kossmann, and Konrad Stocker. The skyline operator. In *ICDE*, pages 421–430, Heidelberg, Germany, 2001.

[BNG+] Yolanda Blanco-Fernández, Martín López Nores, Alberto Gil-Solla, Manuel Ramos Cabrer, Manuela I. Martín-Vicente, and José J. Pazos Arias. Semantic reasoning and mashups : An innovative approach to personalized e-commerce in digital TV. In *SMAP*.

[BRS00] Keith Bradley, Rachael Rafter, and Barry Smyth. Case-based user profiling for content personalisation. In *AH*, pages 62–72, Trento, Italy, 2000.

[BYRN99] Ricardo A. Baeza-Yates and Berthier A. Ribeiro-Neto. *Modern Information Retrieval*. ACM Press / Addison-Wesley, 1999.

[CCDM13] Laurent Candillier, Max Chevalier, Damien Dudognon, and Josiane Mothe. Diversité de recommandations - application à une plateforme de blogs et évaluation. In *CORIA*, pages 269–276, Neuchâtel, Suisse, 2013.

[CCS93] E. F. Codd, S. B. Codd, and C. T. Salley. Providing OLAP (On-Line Analytical Processing) to User-Analysts : An IT Mandate. E. F. Codd and Associates, 1993.

[CD97] Surajit Chaudhuri and Umeshwar Dayal. An overview of data warehousing and olap technology. *Journal of SIGMOD Record*, Volume 26(1) :pages 65–74, 1997.

[CD08] E.A.M. Caron and H.A.M. Daniels. Explanation of exceptional values in multi-dimensional business databases. *European Journal of Operational Research*, Volume 188(3) :pages 884 – 897, 2008.

[CEP09] Gloria Chatzopoulou, Magdalini Eirinaki, and Neoklis Polyzotis. Query recommendations for interactive database exploration. In *SSDBM*, pages 3–18, New Orleans, Louisiana USA, 2009.

[Cho02] Jan Chomicki. Querying with intrinsic preferences. In *EDBT*, pages 34–51, Prague, Czech Republic, 2002.

[Cho03] Jan Chomicki. Preference formulas in relational queries. *ACM Trans. Database Syst.*, 28(4) :427–466, 2003.

[CKK02] Yoon Ho Cho, Jae Kyeong Kim, and Soung Hie Kim. A personalized recommender system based on web usage mining and decision tree induction. *Journal of Expert Systems Applications*, Volume 23(3) :pages 329–342, 2002.

[DBB05] Jérôme Darmont, Omar Boussaid, and Fadila Bentayeb. Dweb : A data warehouse engineering benchmark. In *DaWaK*, pages 85–94, Copenhagen, Denmark, 2005.

[Dev96] Barry Devlin. *Data Warehouse : From Architecture to Implementation.* Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA, 1996.

[DKK05] Jens-Peter Dittrich, Donald Kossmann, and Alexander Kreutz. Bridging the gap between olap and sql. In *VLDB*, pages 1031–1042, Trondheim, Norway, 2005.

[Dou04] Paul Dourish. What we talk about when we talk about context. *Journal of Personal and Ubiquitous Computing*, Volume 8(1) :pages 19–30, 2004.

[DR05] Ian Davidson and S. S. Ravi. Agglomerative hierarchical clustering with constraints : Theoretical and empirical results. In *PKDD*, pages 59–70, Porto, Portugal, 2005.

[DTBC08] Mariam Daoud, Lynda Tamine, Mohand Boughanem, and Bilal Chebaro. Construction des profils utilisateurs à base d'une ontologie pour une recherche d'information personnalisée. In *CORIA*, pages 225–240, Trégastel, France, 2008.

[EVK05] Magdalini Eirinaki, Michalis Vazirgiannis, and Dimitris Kapogiannis. Web path recommendations based on page ranking and markov models. In *WIDM*, pages 2–9, Bremen, Germany, 2005.

[FdS01] João Ferreira and Alberto Rodrigues da Silva. Mysdi : A generic architecture to develop SDI personalised services. In *ICEIS*, pages 262–270, Setúbal, Portugal, 2001.

[FK00] Josef Fink and Alfred Kobsa. A review and analysis of commercial user modeling servers for personalization on the world wide web. *Journal of User Modeling and User-Adapted Interaction*, Volume 10(2-3) :pages 209–249, 2000.

[GBLP96] Jim Gray, Adam Bosworth, Andrew Layman, and Hamid Pirahesh. Data cube : A relational aggregation operator generalizing group-by, cross-tab, and sub-total. In *ICDE*, pages 152–159, New Orleans, Louisiana USA, 1996.

[GG06] Irene Garrigós and Jaime Gómez. Modeling user behaviour aware websites with prml. In *WISM*, pages 1087–1101, Luxemburg, 2006.

[GMN09] Arnaud Giacometti, Patrick Marcel, and Elsa Negre. Recommending multidimensional queries. In *DaWaK*, pages 453–466, Linz, Austria, 2009.

[GMNS09] Arnaud Giacometti, Patrick Marcel, Elsa Negre, and Arnaud Soulet. Query recommendations for olap discovery driven analysis. In *DOLAP*, pages 81–88, Hong Kong, China, 2009.

[GMR98] Matteo Golfarelli, Dario Maio, and Stefano Rizzi. The dimensional fact model : A conceptual model for data warehouses. *Journal of Cooperative Information Systems*, Volume 7(2-3) :pages 215–247, 1998.

[GPMT09] Irene Garrigós, Jesús Pardillo, Jose-Norberto Mazón, and Juan Trujillo. A conceptual modeling approach for olap personalization. In *ER*, pages 401–414, Gramado, Brazil, 2009.

[GR09] Matteo Golfarelli and Stefano Rizzi. Expressing olap preferences. In *SSDBM*, pages 83–91, New Orleans, Louisiana USA, 2009.

[GRB11] Matteo Golfarelli, Stefano Rizzi, and Paolo Biondi. myolap : An approach to express and evaluate olap preferences. *Journal of IEEE Transactions on Knowledge and Data Engineering (TKDE)*, Volume 23(7) :pages 1050–1064, 2011.

[HK00] Jiawei Han and Micheline Kamber. *Data Mining : Concepts and Techniques.* Morgan Kaufmann, 2000.

[HMV99] Carlos A. Hurtado, Alberto O. Mendelzon, and Alejandro A. Vaisman. Maintaining data cubes under dimension updates. In *ICDE*, pages 346–355, Long Beach, California USA, 1999.

[Hoa11] Duong Thi Anh Hoang. Impact analysis for on-demand data warehousing evolution. In *ADBIS*, pages 280–285, Vienna, Austria, 2011.

[HRU96] Venky Harinarayan, Anand Rajaraman, and Jeffrey D. Ullman. Implementing data cubes efficiently. In *International Conference on Management of Data*, pages 205–216, Montreal, Canada, 1996.

[IK05] Yannis E. Ioannidis and Georgia Koutrika. Personalized systems : Models and methods from an ir and db perspective. In *VLDB*, page 1365, Trondheim, Norway, 2005.

[Inm02] W. H. Inmon. *Building the Data Warehouse,3rd Edition.* John Wiley & Sons, Inc., New York, NY, USA, 3rd edition, 2002.

[JRTZ09a] Houssem Jerbi, Franck Ravat, Olivier Teste, and Gilles Zurfluh. Preference-based recommendations for olap analysis. In *DaWaK*, pages 467–478, Linz, Austria, 2009.

[JRTZ09b] Houssem Jerbi, Franck Ravat, Olivier Teste, and Gilles Zurfluh. Preference-based recommendations for olap analysis. In *DaWaK*, pages 467–478, 2009.

[KA08] Norizan M. Kassim and Nor Asiah Abdullah. Customer loyalty in e-commerce settings : An empirical study. *Journal of Electronic Markets*, Volume 18(3) :pages 275–290, 2008.

[KB12] Rym Khemiri and Fadila Bentayeb. Interactive query recommendation assistant. In *DEXA (RSmeetDB)*, pages 93–97, Vienna, Austria, 2012.

[KB13] Rym Khemiri and Fadila Bentayeb. FIMIOQR : Frequent itemsets mining for interactive olap query recommendation. In *DBKDA*, pages 9–14, Seville, Spain, 2013.

[KBB12] Rym Khemiri, Fadila Bentayeb, and Omar Boussaid. Recommandation interactive de requêtes décisionnelles. In *EGC (AIDE)*, pages 92–103, Bordeaux, France, 2012.

[KBG+09] Nodira Khoussainova, Magdalena Balazinska, Wolfgang Gatterbauer, YongChul Kwon, and Dan Suciu. A case for a collaborative query management system. In *CIDR*, Asilomar, California USA, 2009.

[Kel97] Sean Kelly. *Data Warehousing in Action*. John Wiley & Sons, Inc., 1st edition, 1997.

[KI04a] Georgia Koutrika and Yannis E. Ioannidis. Personalization of queries based on user preferences. In *Preferences*, 2004.

[KI04b] Georgia Koutrika and Yannis E. Ioannidis. Personalization of queries in database systems. In *ICDE*, pages 597–608, Shanghai, China, 2004.

[KI05] Georgia Koutrika and Yannis E. Ioannidis. Personalized queries under a generalized preference model. In *ICDE*, Tokyo, Japan, 2005.

[Kie05] Werner Kießling. Preference queries with sv-semantics. In *COMAD*, pages 15–26, Goa, India, 2005.

[KK02] Werner Kießling and Gerhard Köstler. Preference sql - design, implementation, experiences. In *VLDB*, pages 990–1001, Hong Kong, China, 2002.

[KKBS10] Nodira Khoussainova, YongChul Kwon, Magdalena Balazinska, and Dan Suciu. Snipsuggest : Context-aware autocompletion for sql. *Journal of PVLDB*, Volume 4(1) :pages 22–33, 2010.

[KMR+94] Mika Klemettinen, Heikki Mannila, Pirjo Ronkainen, Hannu Toivonen, and A. Inkeri Verkamo. Finding interesting rules from large sets of discovered association rules. In *CIKM*, pages 401–407, Gaithersburg, Maryland USA, 1994.

[KR02] Ralph Kimball and Margy Ross. *The Data Warehouse Toolkit : The Complete Guide to Dimensional Modeling*. John Wiley & Sons, Inc., New York, NY, USA, 2nd edition, 2002.

[LL87] M. Lacroix and Pierre Lavency. Preferences ; putting more knowledge into queries. In *VLDB*, pages 217–225, Brighton, England, 1987.

[Mac67] J. B. MacQueen. Some methods for classification and analysis of multivariate observations. *Proc. of the fifth Berkeley Symposium on Mathematical Statistics and Probability*, Volume 1 :pages 281–297, 1967.

[Mar71] A. Markov. Extension of the Limit Theorems of Probability Theory to a Sum of Variables Connected in a Chain. In R. Howard, editor, *Dynamic Probabilistic Systems (Volume I : Markov Models)*, chapter Appendix B, pages 552–577. John Wiley & Sons, Inc., New York City, 1971.

[MGSG07] Alessandro Micarelli, Fabio Gasparetti, Filippo Sciarrone, and Susan Gauch. The adaptive web. chapter Personalized Search on the World Wide Web, pages 195–230. Springer-Verlag, Berlin, Heidelberg, 2007.

[MS07] Svetlana Mansmann and Marc H. Scholl. Exploring olap aggregates with hierarchical visualization techniques. In *SAC*, pages 1067–1073, 2007.

[MV00] Alberto O. Mendelzon and Alejandro A. Vaisman. Temporal queries in OLAP. In *VLDB 2000, Proceedings of 26th International Conference on Very Large Data Bases, September 10-14, 2000, Cairo, Egypt*, pages 242–253, 2000.

[PBTL99] Nicolas Pasquier, Yves Bastide, Rafik Taouil, and Lotfi Lakhal. Discovering frequent closed itemsets for association rules. In *ICDT*, pages 398–416, 1999.

[PZZ09] Chunhui Piao, Jing Zhao, and Li-Juan Zheng. Research on entropy-based collaborative filtering algorithm and personalized recommendation in e-commerce. *Service Oriented Computing and Applications*, 3(2) :147–157, 2009.

[Riz07] Stefano Rizzi. Olap preferences : a research agenda. In *DOLAP*, pages 99–100, 2007.

[RT08] Franck Ravat and Olivier Teste. *Personalization and OLAP Databases*, volume New Trends in Data Warehousing and Data Analysis, chapter chapter 4, pages 1–22. 2008.

[RTTZ07] Franck Ravat, Olivier Teste, Ronan Tournier, and Gilles Zurfluh. Graphical querying of multidimensional databases. In *ADBIS*, pages 298–313, 2007.

[Sap00] Carsten Sapia. Promise : Predicting query behavior to enable predictive caching strategies for olap systems. In *DaWaK*, pages 224–233, 2000.

[Sar00] Sunita Sarawagi. User-adaptive exploration of multidimensional data. In *VLDB*, pages 307–316, 2000.

[SBdVK13] Alan Said, Alejandro Bellogín, Arjen P. de Vries, and Benjamin Kille. Information retrieval and user-centric recommender system evaluation. In *UMAP*, Rome, Italy, 2013.

[SCDT00] Jaideep Srivastava, Robert Cooley, Mukund Deshpande, and Pang-Ning Tan. Web usage mining : Discovery and applications of usage patterns from web data. *SIGKDD Explorations*, 1(2) :12–23, 2000.

[SDP09] Kostas Stefanidis, Marina Drosou, and Evaggelia Pitoura. "you may also like" results in relational databases. In *PersDB workshop*, 2009.

[SKR99] J. Ben Schafer, Joseph A. Konstan, and John Riedl. Recommender systems in e-commerce. In *EC*, pages 158–166, 1999.

[SLC+08] Yang Sun, Huajing Li, Isaac G. Councill, Jian Huang, Wang-Chien Lee, and C. Lee Giles. Personalized ranking for digital libraries based on log analysis. In *WIDM*, pages 133–140, 2008.

[SM95] Upendra Shardanand and Pattie Maes. Social information filtering : Algorithms for automating "word of mouth". In *CHI*, pages 210–217, Denver, Colorado USA, 1995.

[SN06] Sebastian Stober and Andreas Nürnberger. DAWN - A system for context-based link recommendation in web navigation. In *KES*, pages 763–770, Bournemouth, UK, 2006.

[SP08] Kostas Stefanidis and Evaggelia Pitoura. Fast contextual preference scoring of database tuples. In *EDBT*, pages 344–355, Nantes, France, 2008.

[SS09] Suleyman Salin and Pinar Senkul. Using semantic information for web usage mining based recommendation. In *ISCIS*, pages 236–241, North Cyprus, 2009.

[TS04] Marko Turpeinen and Timo Saari. System architecture for psychological customization of communication technology. In *HICSS*, pages 1–10, Big Island, USA, 2004.

[TSM01] Thomas Thalhammer, Michael Schrefl, and Mukesh K. Mohania. Active data warehouses : complementing olap with analysis rules. *Journal of Data and Knowledge Engineering*, Volume 39(3) :pages 241–269, 2001.

[VCF+07] David Vallet, Pablo Castells, Miriam Fernández, Phivos Mylonas, and Yannis S. Avrithis. Personalized content retrieval in context using ontological knowledge. *IEEE Trans. Circuits and Systems for Video Technology*, Volume 17(3) :pages 336–346, 2007.

[WC00] Kiri Wagstaff and Claire Cardie. Clustering with instance-level constraints. In *ICML*, pages 1103–1110, Stanford, USA, 2000.

[WCRS01] Kiri Wagstaff, Claire Cardie, Seth Rogers, and Stefan Schrödl. Constrained k-means clustering with background knowledge. In *ICML*, pages 577–584, Williamstown, USA, 2001.

[Wid95] Jennifer Widom. Research problems in data warehousing. In *CIKM*, pages 25–30, Baltimore, USA, 1995.

[YPS09] Xiaoyan Yang, Cecilia M. Procopiuc, and Divesh Srivastava. Recommending join queries via query log analysis. In *ICDE*, pages 964–975, Shanghai, China, 2009.

[ZHH02] Jianhan Zhu, Jun Hong, and John G. Hughes. Using markov chains for link prediction in adaptive web sites. In *Soft-Ware*, pages 60–73, Belfast, Northern Ireland, 2002.