REPUBLIQUE ALGERIENNE DEMOCRATIQUE ET POPULAIRE
Ministère de l'Enseignement Supérieur et de la Recherche Scientifique
Université M'hamed Bougara de Boumerdes

Faculté des sciences
Département d'Informatique

MÈMOIRE

présenté par

M^me^ Boucetha Lila née Bousnina

pour obtenir le grade de

MAGISTER

Spécialité : Système informatique et génie des logiciels
Option : *Spécification de Programmes et Traitement de l'Information*

Thème : Personnalisation des requêtes OLAP

Soutenu devant le jury :

Mr Mohamed Mezghiche	Professeur à l'Université de Boumerdes	Président
Mr Ladjel Bellatreche	Professeur à l'Université de Poitiers France	Rapporteur
Mr Kamel Boukhalfa	Maître de Conférences (B) à l'USTHB Alger	Examinateur
Mr Amar Balla	Maître de Conférences (A) à l'ESI Alger	Examinateur
Mr Ahmed Ait Bouziad	Maître de Conférences (B) à l'Université de Boumerdes	Examinateur

Année universitaire : 2009-2010

REMERCIMENTS

En tout premier lieu, je remercie Monsieur *Ladjel Bellatreche*, Professeur à l'université de Poitiers pour avoir dirigé mon travail. Son encadrement, ses critiques constructives, ses précieux conseils m'ont été d'une aide précieuse.

Un grand merci pour :

- Madame Taouri de l'université de Tizi Ouzou ;
- Les étudiants de l'université de Tizi Ouzou : Youcef et Souraya ;
- Monsieur Kamel Aouiche ;
- Mes camarades : Massi, Rbiha et Ikram.

Je tiens à remercier très sincèrement l'ensemble des membres du jury qui me font le grand honneur d'avoir accepté de juger mon travail.

Je remercie aussi tous mes amis et mes collègues de travail qui m'ont assistée et encouragée.

Enfin, je remercie du fond du cœur et avec un grand amour toute ma famille : mes parents, mes sœurs, mes frères et surtout mon mari.

Merci à tous ceux qui de prés ou de loin ont participé à l'aboutissement de ce travail.

RESUME

Les entrepôts de données deviennent des outils indispensables de l'informatique décisionnelle. L'entrepôt est notamment conçu pour supporter des requêtes complexes de décision (requêtes *OLAP*- Online Analytical Processing) dont les résultats sont visualisés sous forme de tableaux croisés. Ces résultats peuvent être très volumineux et souvent ils ne peuvent être visualisés entièrement sur le dispositif d'affichage (*PDA*, téléphone mobile, *etc.*). En conséquence, l'utilisateur doit naviguer longuement afin de trouver l'information pertinente recherchée. L'objectif de ce travail est de personnaliser les requêtes décisionnelles en prenant en compte les profils des utilisateurs. Pour ce faire, chaque utilisateur (décideur) de l'entrepôt de données est invité à donner ses préférences et une contrainte de visualisation contenues dans son profil. Cette contrainte peut être, par exemple, les limitations imposées par le dispositif utilisé pour afficher la réponse d'une requête. Dans cette étude, nous proposons une approche de personnalisation de visualisation : réponse à une requête OLAP sur la base d'un profil utilisateur, contenant une contrainte d'affichage, qui consiste en un couple de nombres entiers représentants le nombre maximal de positions à afficher en ligne et en colonne. L'idée de base de cette approche consiste en la décomposition de ces entiers représentant la contrainte d'affichage en facteurs de nombres premiers, puis l'affectation de ces facteurs aux dimensions concernées par la requête utilisateur selon ses préférences, précisant ainsi le nombre de membres de chaque dimension à afficher. Ces membres sont sélectionnés selon les préférences utilisateur. Le résultat de cette opération est une requête personnalisée dont l'exécution ne contiendra que l'information pertinente pour cet utilisateur. Finalement, une validation de notre proposition en utilisant l'entrepôt de données FoodMart est présentée.

Mots Clés : Entrepôts de données, OLAP, Personnalisation de requêtes, profils, visualisation, préférences utilisateur, contraintes.

ABSTRACT

The data warehouses become essential tools of decisional computing. In particular, it designed to support complex decision queries (*OLAP* queries - Online Analytical Processing), whose results are displayed under the form of cross tables. These results can be very large and often they cannot be visualized entirely on the display device (*PDA*, mobile phone, *etc*.). This is why; the user has to navigate through them to find relevant facts. The aim of this work is to personalize query taking user preferences into account when answering a query. The user (decider) is asked to give his (her) preferences (his profile) and a visualization constraint that can for example, the limitations imposed by the device used to display the answer to a query. In this study, we propose an approach for visualization personalizing: response to an OLAP query, based on a user profile including preferences and constraints. That consists of a couple of integer numbers those represent the maximal number of positions to display on rows and on columns. The basic idea of this approach consists of the decomposition of those integers which represent a constraint in the factors of primary numbers, then the allocation of these factors to the levels concerned by the user query according to its preferences, précising so to speak the number of members for each of the levels to display. These members are selected according to the user preferences. The result of this operation is a personalized query, whose execution will contain only relevant information for this user. Finally, a validation of our proposition, using the data warehouse FoodMart, is presented.

Keywords: Data Warehouse, *OLAP*, query personalization, visualization, profile, user preferences, constraints.

Table des matières

Liste des figures

Liste des Tableaux

Table des acronymes

Acronyme	***Signification***
BCNF	Boyce-Codd Normal Form
BDD	Base De Données
CQP	Constraint of Query Personalization
ETL	Extract-Transform-Load
HTML	Hyper Text Markup language
HTTP	Hyper Text Transfert Protocol
IHM	Interface Homme Machine
JDBC	Java DataBase Connectivity
JSP	Java Servers Pages
MDX	Multi Dimensional Expressions)
MySQL	My Structured Query Language
OLAP	On Line Analytical Processing
OLTP	On Line Transaction Processing
PDA	Personal Digital Assistant
RI	Recherche d'Information
SGBD	Système de Gestion de Bases de Données
SQL	Structered Query Langage
WAP	Wirless Application Protocol
WML	Wireless Markup Language
XML	eXtensible Markup Language

Introduction générale

Les systèmes de recherche d'information sont des outils qui ont permis, jusqu'à aujourd'hui d'améliorer la qualité des services d'accès à l'information. Au fur et à mesure que le volume de données s'accroît et les informations se diversifient, ces systèmes (moteur web, SGBD….) délivrent des résultats massifs, générant ainsi une surcharge informationnelle en réponse aux requêtes des utilisateurs dans laquelle il est difficile de distinguer l'information pertinente d'une information secondaire. Donc le problème n'est pas dans la disponibilité de l'information mais dans sa pertinence relative aux besoins précis de l'utilisateur, c'est pourquoi les travaux s'orientent actuellement vers la révision de cycle de vie d'une requête dans la perspective d'intégrer l'utilisateur comme composante du modèle global de recherche et ce, dans le but de lui délivrer une information pertinente adaptée à ses besoins et préférences précis. Ces travaux s'inscrivent dans le cadre de la personnalisation de l'information qui est vue comme l'une des solutions pouvant maintenir le web comme une ressource d'information viable.

La personnalisation de l'information constitue un enjeu majeur pour l'industrie informatique, son but est de faciliter l'expression du besoin de l'utilisateur et de lui permettre d'obtenir des informations pertinentes lors de ses interactions avec un système d'information. La pertinence de l'information délivrée et son adaptation aux préférences des utilisateurs sont des facteurs clés du succès ou du rejet d'un tel système. Pour rendre la personnalisation effective, on a besoin de collecter et sauvegarder l'ensemble des critères et des préférences personnalisables spécifiques à chaque utilisateur. Ces données sont souvent regroupées sous forme de *profil*. Le contenu du profil d'un utilisateur peut varier selon les approches et les applications. Le profil de l'utilisateur peut être construit explicitement ou implicitement. Dans l'approche explicite, l'utilisateur fourni lui-même les informations le concernant (données personnelles, préférences, etc.). Dans l'approche implicite, les informations du profil sont acquises par des techniques d'apprentissage ou de datamining exploitant les historiques des actions et des choix passés des utilisateurs.

Du point de vue recherche, la personnalisation de l'information a été abordée principalement par la communauté Recherche d'Information (RI), la communauté Bases de Données (BDD) et la communauté Interface Homme Machine (IHM) et récemment

[Bouzeghoub et al, 2005] par la communauté des Entrepôts de Données (ED). Bien que les deux premiers domaines sont très proches dans leur objectif final (accès efficace à l'information) et qui ont le même but (personnalisation des requêtes), les deux domaines de recherche se distinguent généralement par la nature de l'information traitée (documents textuels pour la RI et tables structurées pour les BDD) et le mode d'accès à cette information (accès par mots-clés plus ou moins complexes et pas à pas pour la RI face à un accès par expressions logiques et de façon globale pour les BDD et les ED).
Dans le domaine des entrepôts de données, la personnalisation n'a été abordée que récemment et notre contribution se situe dans ce cadre.

Nous nous intéressons à la personnalisation de l'information dans le contexte de l'interrogation des entrepôts de données par requêtes OLAP (On-Line Analytical Processing) qui permettent l'analyse interactive d'un gros volume de données multidimensionnelles à l'aide d'opérateurs spécifiques de

(a) consolidation et agrégation pour résumer l'information et de (b) présentation et structuration pour naviguer et explorer l'information.

Problématique

Les utilisateurs finaux des outils d'aide à la décision comptent fortement sur la visualisation des réponses à des requêtes de leurs analyses interactives, sur des quantités importantes de données. Très souvent, ces réponses ne peuvent pas être visualisées entièrement sur le dispositif d'affichage (écran d'ordinateur, téléphone portable, PDA, etc.). En conséquence, l'utilisateur doit naviguer longuement afin de trouver l'information recherchée. Ceci car, les systèmes OLAP actuels ont peu de connaissances sur l'utilisateur. Ils ne tiennent pas compte des caractéristiques spécifiques de chaque utilisateur pour la restitution des données, à savoir ses objectifs et ses centres d'intérêts. Ceci oblige l'analyste à naviguer au sein des données par un enchainement d'opérations et une succession de résultats intermédiaires pour obtenir les données pertinentes à sa prise de décision (adaptées à ses besoins spécifiques d'analyse). L'analyse OLAP peut s'avérer alors une tâche fastidieuse qui dégrade les performances du processus d'analyse décisionnelle. Cette dégradation est aggravée par un coût d'exécution important des requêtes dans un environnement OLAP avec un grand nombre de dimensions. Notre objectif est de personnaliser les requêtes décisionnelles interrogeant des bases de données multidimensionnelles en restituant les données en fonction des préférences utilisateur et de ses contraintes. Pour ce faire, chaque utilisateur (décideur) de l'entrepôt de données est invité à donner ses préférences (son profil)

et une contrainte de visualisation. Cette contrainte peut être, par exemple, les limitations imposées par le dispositif utilisé pour afficher la réponse d'une requête.

Ce mémoire est composé des parties suivantes dont le contenu est exposé brièvement ci-dessous :

La partie 1 : représente l'état de l'art, elle comporte :

Le chapitre 2 : Personnalisation de l'information et le profil utilisateur

Ce chapitre présente le contexte d'étude, à savoir, la notion de personnalisation et ses domaines ainsi que la notion de profil utilisateur dans laquelle nous avons parlé de processus de la modélisation du profil utilisateur.

Le chapitre 3 : La personnalisation de l'information dans le domaine des bases de données

Ce chapitre constitue une initiation à la personnalisation, il aborde la notion de préférence, les techniques de reformulation des requêtes et les contraintes d'optimalisation qui doivent être prise en considération lors de l'exécution de la requête.

Le chapitre 4 : La personnalisation et les entrepôts de données

Dans ce chapitre, nous avons présenté d'abord les concepts généraux manipulés dans les entrepôts de données, puis nous avons dressé un panorama des travaux réalisés sur la personnalisation dans les entrepôts de données en général. Puis nous avons étudié de manière détaillée la personnalisation des visualisations : réponses à des requêtes OLAP.

La partie 2 : Présente la personnalisation des visualisations dans les entrepôts de données

Nous détaillons dans cette partie notre approche de personnalisation sur des requêtes OLAP, portant sur la visualisation des réponses à ces requêtes.

La partie 3 : Décrit la validation de notre contribution

La dernière partie de ce travail traite la réalisation effectuée présentée dans le chapitre 6. Nous présentons le prototype Mob_OLAP pour les applications mobiles, en décrivant tout d'abord les outils et l'environnement avec lesquels ce système a été réalisé. Puis, nous décrivons son architecture, ainsi que son fonctionnement. Nous donnons ensuite les résultats des expérimentations. Cette application peut être une contribution pour simplifier l'accès à une information pertinente en tenant compte des contraintes du dispositif utilisé.

Enfin, nous conclurons en exposant les caractéristiques du système, et des perspectives sont ensuite suggérées.

PREMIERE PARTIE

ETAT DE L'ART

La première partie de ce mémoire représente un état de l'art complet sur le problème de la personnalisation de l'information d'une manière générale et les données (provenant d'un entrepôt de données ou d'une base de données) en particulier. Cette partie est présentée selon trois chapitres distincts.

Dans le chapitre 2, nous visons à situer d'abord le contexte de d'étude qui est la personnalisation et les domaines d'études : commerce électronique, apprentissage assistée par ordinateur, etc. Puis nous avons présenté la notion de profil utilisateur dans laquelle nous avons expliqué le processus de la modélisation du profil utilisateur.

Le chapitre 3 est consacré à la personnalisation dans les bases de données relationnelles qui est un domaine très proche à celui des entrepôts de données. Dans ce chapitre, nous avons abordé la notion de préférence, les techniques de reformulation des requêtes et les contraintes d'optimalisation qui doivent être prise en considération lors de l'exécution de la requête.

Le chapitre 4 présente le problème de la personnalisation des entrepôts de données. Cette personnalisation concerne les requêtes OLAP (On Line Analytical Processing). Ce chapitre introduit d'abord les concepts généraux des entrepôts, les différentes implémentations ROLAP et les opérateurs OLAP liés à la structure du cube de données modélisant un entrepôt de données. Puis nous avons dressé un panorama sur les travaux récents sur la personnalisation et la recommandation des requêtes décisionnelles dans les entrepôts de données. Enfin un état de l'art sur les travaux de personnalisation des visualisations est présenté.

2 Personnalisation de l'information et le profil utilisateur

1. La personnalisation de l'information

Les modèles classiques de recherche d'information sont fondés sur la supposition naïve que l'utilisateur est complètement représenté par sa requête. Ceci a pour corollaire que deux utilisateurs *différents* qui posent la même requête auront exactement les mêmes résultats.

Les premières solutions apportées à ce type de problèmes peuvent s'apparenter à la personnalisation de l'information.

1.1. La notion de personnalisation

Définitions :

1- La personnalisation de l'information est une dimension qui permet la mise en œuvre d'un système centré utilisateur non dans le sens d'un utilisateur générique mais d'un utilisateur spécifique [Tamine et al, 2007].

2- La personnalisation de l'information se définit, entre autres, par un ensemble de préférences individuelles représentées par des couples (attribut, valeur), par des ordonnancements de critères ou par des règles sémantiques spécifiques à chaque utilisateur ou communauté d'utilisateurs. Ces modes de spécification servent à décrire le centre d'intérêt de l'utilisateur, le niveau de qualité des données qu'il désire ou des modalités de présentation de ces données [Bouzeghoub et al, 2005].

1.2. Les Domaines de la personnalisation

Parmi les domaines qui font appel à la personnalisation on peut citer les suivants :

1.2.1. Les applications

- **Commerce électronique**

Ce domaine d'application recouvre la vente de produits de toute nature y compris les contenus multimédias « à la demande » pour lesquels la démarche commerciale tire un grand profit des techniques de personnalisation. Les apports de la personnalisation s'échelonnent de la mémorisation des données personnelles pour éviter leur re-saisie systématique jusqu'à la recommandation de produit basée sur les achats précédents et/ou le comportement de l'utilisateur sur le site web de vente.

- **La dissémination sélective d'information**

Concerne la diffusion d'information culturelle et d'actualité, la personnalisation permet le filtrage des informations en tenant compte d'un profil traduisant le centre d'intérêt, la langue, la religion et la position géographique de l'utilisateur.

- **Apprentissage assisté par ordinateur**

Ce domaine d'application concerne tous les environnements informatiques pour l'apprentissage humain ou de veille technologique. La personnalisation permet de définir des objectifs et des formations sur mesure (selon les connaissances, le style d'apprentissage préféré, etc.) et de suivre l'apprenant aux cours de sa formation afin d'adapter la réaction du système d'apprentissage à son état d'avancement et à son comportement.

- **Accès aux bibliothèques électroniques**

La personnalisation s'applique à différents niveaux :

- Limiter l'accès aux seuls documents auxquels l'abonné a souscrit.
- Guider la navigation de l'utilisateur au sein de ces documents selon sa requête du moment et recommander les nouveautés en fonction du profil.

- **Configuration des logiciels (réseau, composantes)**

La technologie informatique elle-même fait appel de plus en plus aux techniques de la personnalisation telle que la configuration de systèmes d'exploitation, de protocole réseaux ou de services web en fonction des besoins des utilisateurs.

- **Systèmes d'informations mobiles**

Les services accessibles via les téléphones mobiles et les systèmes embarqués requièrent d'un coté une personnalisation leur permettant de limiter l'effort et le temps passé à

la recherche de l'information pertinente, et une adaptabilité aux contraintes physiques ou techniques (écran de petite taille, clavier absent ou réduit, bande passante limitée, etc.).
Les vendeurs de service présentent souvent ces techniques comme des moyens d'aide à la décision pour l'utilisateur.

1.2.2. Les technologies

La personnalisation est implémentée dans les interfaces homme/machine (IHM), les moteurs de recherche (moteur web), les systèmes de gestion de bases de données SGBD et dans les serveurs OLAP.
La personnalisation telle qu'elle était abordée dans les deux derniers systèmes sera détaillée respectivement dans les chapitres 3 et 4.

2. Le profil utilisateur

La notion du profil utilisateur est apparue vers les années 80 avec les assistants et les agents d'interface. Cette apparition est due principalement au besoin de créer des applications personnalisées capables de s'adapter à l'utilisateur [Kobs 2001].

Personnaliser une application pour un utilisateur particulier nécessite de disposer d'informations sur ce dernier, ce qui permettra d'évaluer la pertinence des objets disponibles ou d'aider le système à faire des choix. Le modèle utilisateur est « une source de connaissances ou une base de données sur un utilisateur » [McTear, 1993].

Dans ce qui suit nous allons présenter le profil utilisateur tel qu'il était abordé surtout par la communauté des BDD et ED.

2.1. La notion de profil

Définitions :

*1. Un profil utilisateur est une collection d'informations sur l'utilisateur. Cette collection peut être vue comme un ensemble de caractéristiques avec des valeurs associées contenant par exemple ce que l'utilisateur préfère, ce qu'il est capable de faire...etc. On doit également prendre en compte l'historique des actions de l'utilisateur, voir leur évolution dans le temps. [web1]***.**

2. Un profil utilisateur regroupe l'ensemble des connaissances nécessaires à une évaluation efficace des requêtes et à une production d'une information pertinente adaptée à chaque utilisateur [Bouzeghoub et al, 2004].

2.2. Le contenu du profil

Dans un processus d'accès à des informations structurées comme les bases de données ou les entrepôts de données, un profil utilisateur peut être composé de : **(1)** préférences sur le contenu de la réponse à la requête, **(2)** préférences sur la présentation de la réponse, et/ou **(3)** des conditions d'exploitation (les conditions matérielles et/ou les contraintes utilisateur). Nous présentons dans les chapitres suivants le détail de ces composants spécifiques à chaque domaine.

2.3. Modélisation du profil utilisateur

L'introduction de la dimension utilisateur dans un processus d'accès à l'information, mérite, voire nécessite une réflexion sur la modélisation de l'entité *utilisateur*.

Différents travaux ont considéré que la modélisation de l'utilisateur; dans le cadre d'un système d'accès personnalisé à l'information, est un processus caractérisé par trois phases :

1. La première porte sur la définition d'une *représentation* des unités d'information caractérisant l'utilisateur du système. Elle correspond à la définition de la structure du profil utilisateur présentée dans la section précédente,
2. La deuxième phase est liée à *l'instanciation* de cette représentation au cours d'une activité d'accès à l'information pour un utilisateur particulier. Elle regroupe des techniques d'acquisition des données utilisateur ainsi que des approches de construction pour agencer les informations collectées selon la structure représentative définie lors de l'étape précédente,
3. Enfin, la troisième phase concerne *l'évolution* du profil au cours du temps. Elle nécessite la mise en place de stratégies de mise à jour du contenu informationnel du profil.

Dans ce qui suit nous allons exposer les différentes approches utilisées pour la mise en œuvre de chaque étape.

Approches de représentation de profil

Il existe principalement trois types de représentation [Tamine et al, 2007]:

1. ***Ensembliste*** : le profil y est généralement formalisé comme des ensembles d'éléments pondérés, cette approche a été largement utilisée dans le domaine de la RI.
2. ***Sémantique*** : la représentation du profil met en évidence, dans ce cas, les relations sémantiques entre informations le contenant. La représentation est essentiellement basée sur l'utilisation d'ontologies ou réseaux sémantiques probabilistes ce volet n'est pas encore abordé dans le contexte des entrepôts de données,

3. ***Multidimensionnelle*** : le profil est structuré selon un ensemble de dimensions, représentées selon divers formalismes. Vu la nature des informations caractérisant les systèmes d'interrogation des données provenant d'un entrepôt, cette approche a été adoptée pour la présentation de notre modèle.

2.4. Construction du profil

La construction du profil traduit un processus qui permet d'instancier sa représentation. Elle s'effectue en deux étapes : **(1)** l'acquisition et la collecte des données utilisateur ; **(2)** puis la construction proprement dite du profil [Tamine et al, 2008].

La première phase consiste à collecter les informations pertinentes pour instancier le profil de l'utilisateur. Ce processus peut collecter ces informations soit directement à partir de la machine de l'utilisateur (côté client) ou à partir de l'application (côté serveur). Ce processus d'acquisition peut être explicite et/ou implicite :

i. L'approche explicite consiste à obtenir les informations directement de l'utilisateur,

ii. L'approche implicite, largement motivée par les travaux actuels dans le domaine, implique l'exploitation des données de comportement de l'utilisateur pour inférer son profil.

a) L'acquisition explicite

Cette technique constitue une approche simple pour obtenir des informations sur l'utilisateur. On interroge directement l'utilisateur ou on lui demande par exemple de remplir des formulaires pour collecter ses préférences sur les dimensions, les membres ainsi que des informations décrivant son environnement à savoir la taille de son écran, la vitesse de son processeur et la taille de sa mémoire, etc.

En effet, l'utilisateur émet directement son jugement d'intérêt en donnant une valeur de pertinence sur une échelle graduée allant du moins intéressant au plus intéressant.

b) L'acquisition implicite (apprentissage)

L'acquisition implicite ou « feedback implicite » consiste à collecter les informations décrivant l'utilisateur, en observant les dimensions et les membres fréquemment sollicités et en scrutant les caractéristiques de l'environnement à partir duquel il intervient (les capacités et les limites du dispositif utilisé lors de ces interactions). Et ce, en se basant sur l'**historique** de ses interactions avec le système.

Le principal avantage de cette approche est qu'elle ne nécessite aucune implication directe de l'utilisateur, ni de temps passé à émettre des jugements, ni un effort d'attention particulier lors de son interrogation. On peut noter aussi que la sécurité et la confidentialité

des informations (par exemple le schéma de l'entrepôt) sont préservées. L'inconvénient se trouve dans la complexité des algorithmes utilisés qui nécessitent beaucoup de temps. Des exemples de telles techniques sont les réseaux de neurones, les méthodes de classification (raisonnement par cas, classificateurs bayésiens…), les règles d'association.

Bien qu'ils existent déjà plusieurs techniques permettant de capturer les préférences d'un utilisateur, leur utilisation n'est pas encore bien comprise pour permettre la construction et la mise à jour de son profil de façon automatique et complètement transparente pour lui. Actuellement nous pouvons espérer que ceci se fasse avec une implication minimale de l'utilisateur. Le principe est que l'utilisateur doit avoir le contrôle de son profil à tout moment afin de pouvoir invalider les mises à jour incorrectes du profil. Après une telle invalidation, le gestionnaire doit prendre en compte et modifier la manière (le processus) de gestion et de mise à jour du profil.

2.5. Exploitation de profil

Nous abordons dans cette section la problématique liée à : à quelle étape de cycle de vie de la requête le profil est intégré ?

Le contenu du profil d'un utilisateur peut être utilisé à différents moments de cycle de vie de la requête. Il peut servir pour enrichir le contenu de la requête, ou pour optimiser son exécution ou encore être utilisé pour adapter les résultats selon les modalités de présentation. En général, il peut influer sur l'exécution de la requête sans nécessairement intervenir dans l'expression de celle-ci.

La figure 2.1 illustre les principales phases du processus d'accès personnalisé à l'information selon le cycle de vie de la requête où l'on prend en compte le profil utilisateur, à savoir :

(1) La phase d'exécution de la requête : lors de cette phase le contenu informationnel du profil peut être exploité directement dans le processus d'exécution de la requête. L'intégration du profil lors de la phase d'exécution de la requête est beaucoup plus utilisée dans les systèmes de recherche d'information (RI).

(2) La phase de présentation des résultats : La personnalisation à ce stade du processus d'accès à l'information offre une solution en réordonnant les résultats pour ne présenter à l'utilisateur que les informations pertinentes en réponse à son besoin en information. Ce besoin est formulé en conjuguant les informations données par l'utilisateur (la requête soumise) et celles extraites de son profil. Ainsi, la restitution des résultats s'effectue en fonction de la notion de pertinence personnelle de l'utilisateur. Dans le contexte des entrepôts

de données, la personnalisation consiste à exploiter les informations du profil utilisateur sur l'aspect visualisation qui est primordial dans le contexte de l'analyse en ligne. Ce type de personnalisation sera détaillé dans le chapitre 4, section 2. 2. 5.

(3) La phase d'affinement de la requête : Les requêtes utilisateur sont assurément une source évidente importante pour l'identification des besoins en information de l'utilisateur. Néanmoins, les utilisateurs soumettent souvent des requêtes très courtes et ambiguës. L'objectif de la personnalisation à ce stade du cycle de vie de la requête est de clarifier le besoin en information de l'utilisateur en se basant sur son profil. Ainsi, la reformulation de requête dans ce cadre intègre les composantes informationnelles issues du profil de l'utilisateur pour identifier, enrichir et cibler son intention.

Dans le cadre d'un accès personnalisé aux bases de données, l'approche de reformulation de requête consiste à proposer des algorithmes d'enrichissement de la requête utilisateur par des prédicats de préférence candidats choisis du profil. Ce processus sera détaillé dans le chapitre suivant, section 4.1.

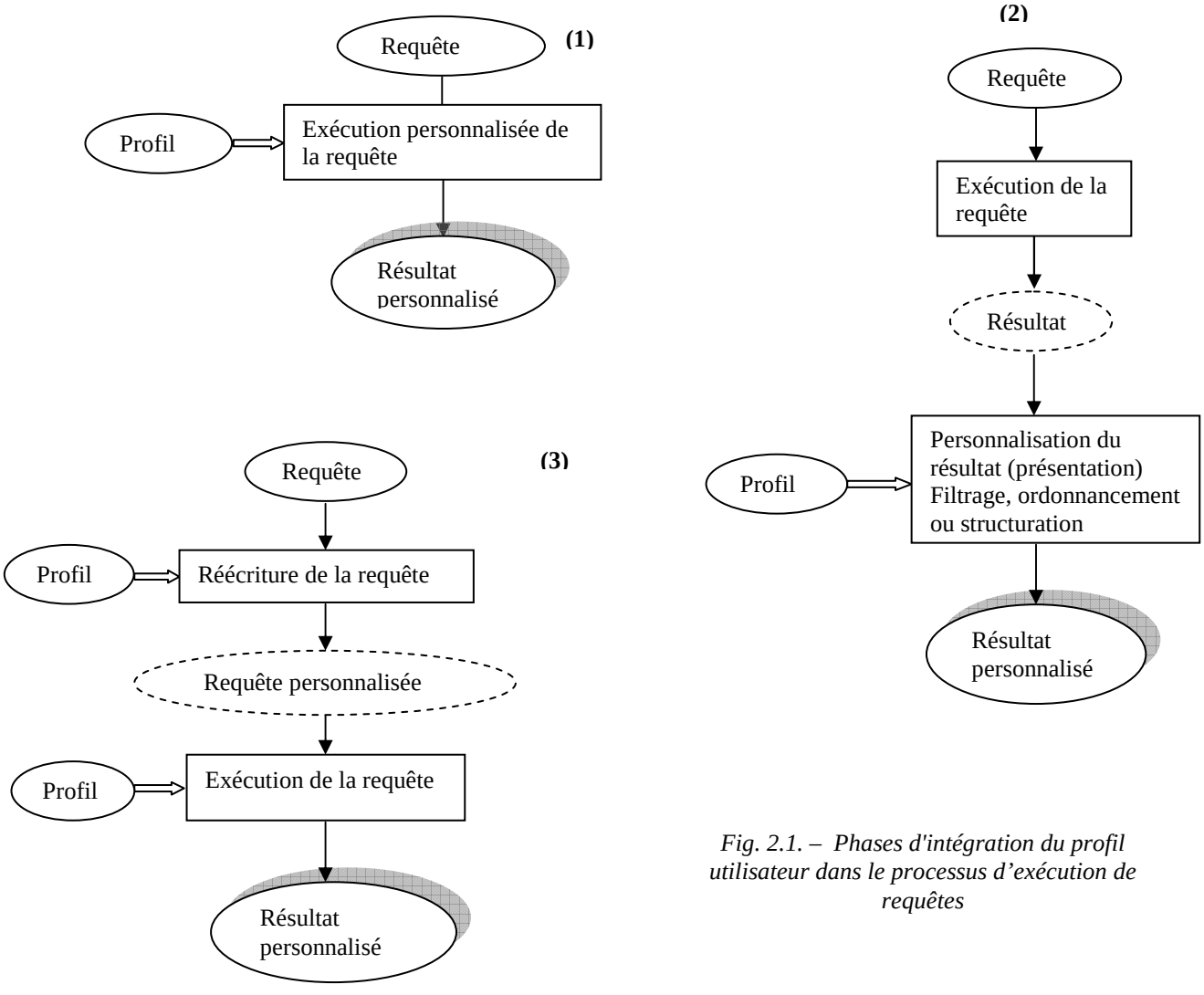

Fig. 2.1. – Phases d'intégration du profil utilisateur dans le processus d'exécution de requêtes

2.6. Évolution du profil utilisateur

L'évolution des profils désigne leur adaptation à la variation des préférences des utilisateurs et au changement de leur environnement d'interaction, et par conséquent, de leurs besoins en information au cours du temps.

La technique utilisée pour faire évolué le profil est inhérente à celle de la construction. Néanmoins, adapter le profil utilisateur implique des changements au niveau de contenu qui conduisent éventuellement à la suppression de quelques dimensions ou à l'émergence d'autres dimensions avec l'augmentation ou la diminution de l'intérêt.

Si l'acquisition du profil est implicite et afin de faire face aux changements d'intérêts dynamiques, le système doit non seulement dépister des intérêts dérivant de l'utilisateur afin d'identifier de nouveaux intérêts émergeants, mais également inférer à chaque connexion les caractéristiques de son environnement à savoir la taille de son écran, la capacité de sa mémoire, etc.

A notre connaissance même en RI peu de travaux ont abordé cet aspect d'évolution du profil. En effet, l'évolution n'est prise en considération que dans la mesure où le profil a une existence pérenne, ce qui n'est pas le cas dans la majorité des systèmes car, le plus souvent à chaque connexion de l'utilisateur, un nouveau profil est instancié.

3. Les systèmes d'accès personnalisé à l'information

Définitions :

Un système d'accès personnalisé à l'information est un système qui intègre l'utilisateur, en tant que structure informationnelle, tout au long de la chaîne d'accès à l'information.

4. Architecture d'un système d'accès personnalisé à l'information

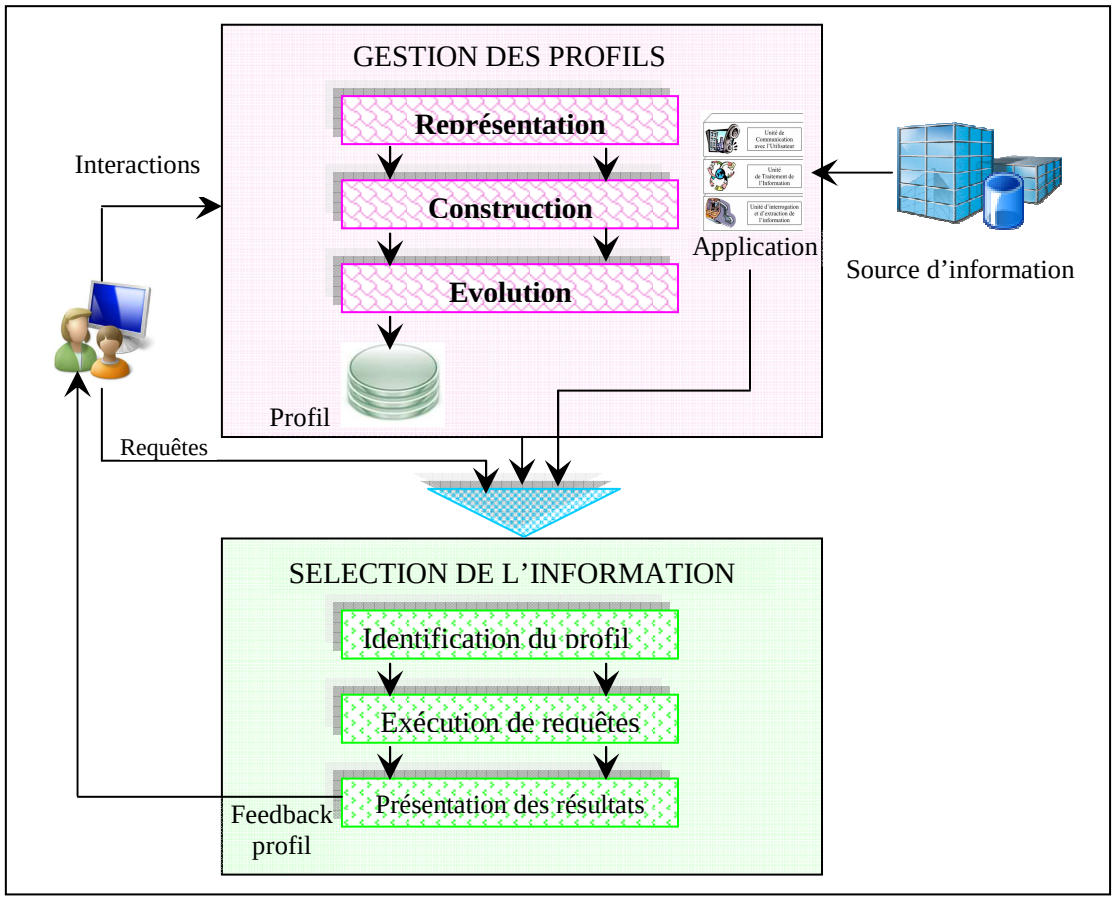

Fig. 2.2 – Architecture fonctionnelle d'un système d'accès personnalisé à l'information [Lechani et al, 2005].

Cette architecture est centrée autour de l'utilisateur, sur la base de cette architecture nous dégageons principalement deux fonctions fondamentales qui sont la gestion des profils et la sélection de l'information.

5. Conclusion

A travers ce chapitre, nous avons constaté que la personnalisation est une approche capitale dans le développement des systèmes futurs, nous définissions la notion de profil utilisateur, les données contenues dans le profil doivent être correctes et justes car elles ont un impact direct sur les résultats qui vont être obtenus. Le fait que l'utilisateur a souvent des idées floues sur ses préférences, complique l'extraction des caractéristiques pertinentes et nécessite un processus de découverte de ses préférences. Ce processus doit être fait avec l'implication minimale du coté du client et en même temps doit fournir des données correctes par rapport à l'utilisateur. La construction et la mise à jour du profil sont des problèmes qui ne rentrent pas dans le cadre de nos objectifs.

Par contre, le contenu du profil d'un utilisateur de bases de données sera abordé dans le prochain chapitre décrivant la personnalisation dans le domaine des BDD. Et le profil d'un décideur sera détaillé dans le chapitre 4.

3 La personnalisation dans le domaine des BDD

Actuellement, les chercheurs travaillent sur la manière de faciliter l'expression du besoin de l'utilisateur et de lui permettre d'obtenir des informations pertinentes lors de ses accès aux systèmes d'information en modélisant le profil utilisateur**.** Ils travaillent aussi à l'adaptation des systèmes de base de données aux différents utilisateurs qui n'ont pas une idée précise sur l'information qu'ils recherchent c'est ce qu'ils appellent accès personnalisé à l'information.

Dans le cas des bases de données (BDD), un utilisateur accède à l'instance de base de données à l'aide d'une requête dont la réponse est composée d'un ensemble de tuples.

Dans ce cadre, la personnalisation consiste à prolonger le langage d'interrogation et la réécriture des requêtes.

Avant d'introduire le cadre théorique dans lequel s'inscrit notre travail, nous nous proposons, dans ce chapitre, une initiation à la personnalisation en définissant tout d'abord le système personnalisé de base de données ainsi que la notion de préférence, auxquelles nous ferons appel tout au long de ce document, et en détaillant, par la suite, le modèle de profil, les technique de reformulation des requêtes. Enfin, nous définissons les contraintes d'optimisation des requêtes personnalisées.

1. Architecture d'un système personnalisé de base de données

L'architecture générale d'un système de base de données personnalisé est basée essentiellement sur :

- **La création de profil** : Qui s'occupe de la représentation des caractéristiques d'utilisateur sous forme d'un *modèle de profil* qui constitue le noyau central de la personnalisation.

- **La personnalisation de la requête :** Qui s'occupe de l'intégration des informations du profil à la requête entrante afin de sélectionner le contenu pertinent qui est les tuples ou les attributs;
- **La personnalisation de présentation :** Qui s'occupe de la présentation de résultat d'une manière adaptative selon les préférences de l'utilisateur. Ce type de personnalisation n'a pas été beaucoup traité dans les BDD. Car la sélection (clause select dans la requête) (1) permet de déterminer les attributs utiles à afficher, et (2) l'ordonnancement permet d'ordonner les attributs sélectionnés par ordre décroissant de leur utilité à l'affichage.

Par contre, dans les entrepôts de données, la personnalisation traite cet aspect : la présentation ou visualisation

La figure suivante illustre cette architecture :

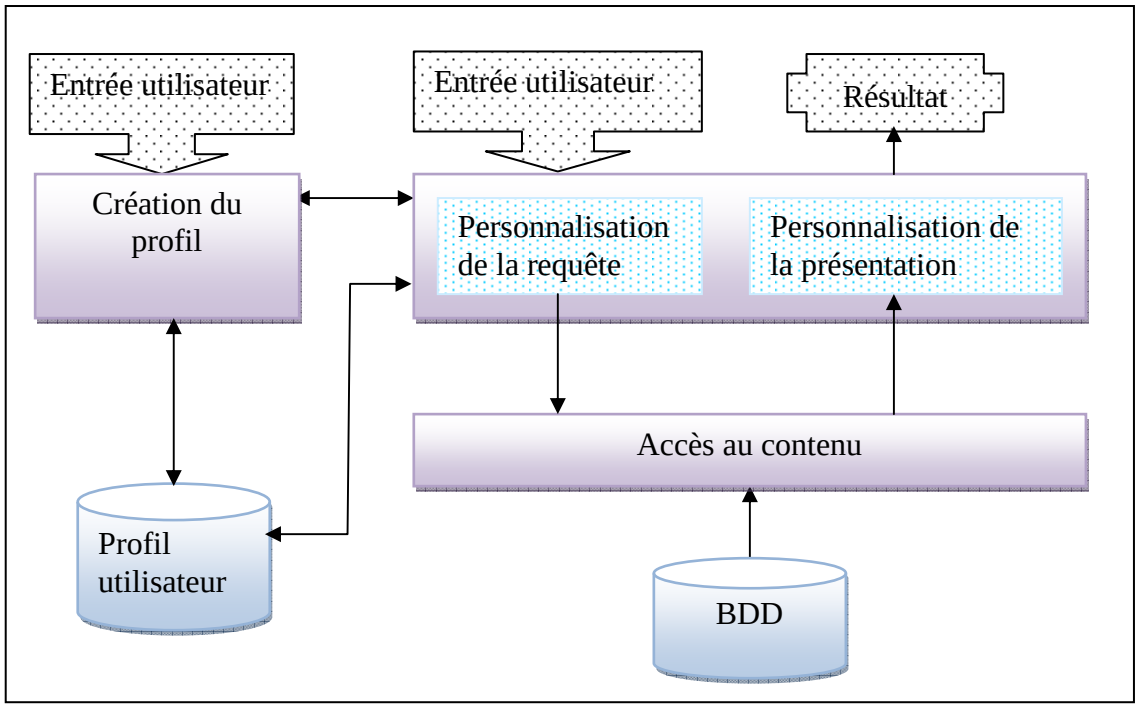

Fig. 3.1 - Architecture d'un système de base de données personnalisé [Koutrika et Ioannidis 2004].

2. La notion de préférences

Définition 1 :

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.

Dans le contexte des BDD, les préférences utilisateur permettent d'ordonner les tuples de la réponse selon leur importance et ainsi de déterminer quels sont les tuples les plus intéressants.

Les préférences dans ce domaine sont exprimées sur le contenu (attributs, tuples ou requêtes)

2. 1. L'expression des préférences

Ils existent plusieurs approches de classification des préférences. Selon le type de données utilisées pour la comparaison de deux éléments on distingue des préférences intrinsèques et extrinsèques.

- ***Les préférences intrinsèques*** : portent uniquement sur les valeurs des attributs d'un élément.
- ***Les préférences extrinsèques*** : prennent en compte des facteurs extérieurs comme l'origine d'un élément.

Une autre manière de classer les approches de personnalisation est basée sur la manière de comparer deux éléments. On retrouve ici :

- ***L'approche qualitative*** où la préférence est spécifiée directement par des relations binaires de préférence et
- ***L'approche quantitative*** où les préférences sont exprimées indirectement par le biais de fonctions qui attribuent un nombre à chaque élément et la pertinence d'un élément dépend du nombre attribué.

2. 1. 1. Préférences qualitatives

Avant de présenter les préférences qualitatives dans les BDD, nous allons introduire les notions utilisées dans ce domaine.

Dans les BDD, les données manipulées sont représentées par un type de structure appelé relation. Autour de cette notion, on trouve les éléments suivants [Abiteboul et al., 1995]:

- **Domaine** : ensemble de constantes défini en extension ou en intention.
- **Produit cartésien** : le produit cartésien d'un ensemble de domaines $D_1, D_2, \ldots, D_k$, noté $D_1 \times \ldots \times D_k$, est l'ensemble de k-uplets (ou tuples) $\langle t_1, \ldots, t_k \rangle$ tels que $t_i \in D_i, i \in [1, k]$.
- **Relation** : sous-ensemble de produit cartésien de k domaines ($k > 0$). Si on note R cette relation alors $R \subseteq D_1 \times \ldots \times D_k$.

Une relation est représentée par une table à deux dimensions. Chaque ligne correspond à un tuple et chaque colonne correspond à un domaine. Le nom donné à une colonne de la table est un attribut.

- **Instance d'une relation** : ensemble des tuples d'une relation qui existent dans la base de données à un instant donné.
- **Schéma d'une relation** : un schéma d'une relation de nom R est noté R(A) où $A = \{A_1, A_2, \ldots, A_k\}$ est un ensemble d'attributs. Il y a un domaine $dom(A_i)$ pour chaque attribut A_i, $i \in [1, k]$.

Définition 3.1 - Relation de préférence sur une relation. *Une relation de préférence* $>_R$ *sur une relation de schéma R(A) est une relation binaire sur* $\times^k{}_{i=1} dom(A_i)$.

***Exemple 3.1* :** Soit la table ci-dessous décrivant une instance de la relation Car. Le schéma de cette relation est Car (Make, Model, Color, Price, Year).

Car	*Make*	*Model*	*Color*	*Price*	*Year*	
	BMW	330	Black	100k	1995	t_1
	Ford	Escort	White	20K	1997	t_2
	Ford	Escort	Yellow	12K	1990	t_3
	Ford	Escort	White	18K	1995	t_4
	Toyota	Corolla	Silver	15K	2003	t_5
	Ferrari	360	Red	100K	1995	t_6
	BMW	360	Black	100K	1995	t_7

Tab. 3.1 – Table Car

Un utilisateur peut exprimer un ensemble de préférences sur la relation de l'exemple 3.1 par : $t_1 >_R t_2$, $t_3 >_R t_2$, $t_3 >_R t_1$, $t_5 >_R t_6$, $t_7 \approx_R t_6$, et $t_2 \|_R t_5$.

Sachant que : $\approx_R$: est la relation d'indifférence et $\|_R$: est la relation d'incomparabilité.

Cela signifie que :

- t_1 est strictement préféré à t_2
- t_7 et t_6 sont également préférés
- Il n'y a ni indifférence, ni préférence entre t_2 et t_5 et on dit que t_2 et t_5 ne sont pas comparables

2. 1. 2. Préférences quantitatives

Définition 3.2 (- Fonction d'utilité sur une relation). *Étant donnée une relation de schéma R(A), une fonction d'utilité u sur R(A) est une fonction définie de* $\times^k{}_{i=1} dom(A_i)$ *vers [0, 1].*

***Exemple 3.2* :** Un utilisateur peut fournir les scores numériques suivants sur les tuples de la relation de l'exemple 3.1 : $u(t_1) = 0.8$, $u(t_2) = 0.7$, $u(t_3) = 0.9$, $u(t_5) = 0.6$, etc. Dans ce cas, on a par exemple $t_1 >_u t_2$ car $u(t_1) = 0.8 > u(t_2) = 0.7$.

On note aussi que dans cette représentation par une fonction d'utilité, deux éléments de même utilité sont considérés comme indifférents.

***Note** : L'approche qualitative est plus générale que l'approche quantitative car elle peut définir les relations de préférences sur des fonctions numériques si elles sont données explicitement, tandis que toutes les relations de préférences ne peuvent pas être capturées par des fonctions numériques.*

Étant donné un ensemble d'éléments E, une relation de préférence $>_R$ sur E peut être définie en extension en donnant la liste des couples $(x, y) \in E^2$ tels que $x >_R y$. Mais, cette technique n'est pas réaliste lorsque le nombre d'éléments est important.

Dans ce qui suit, on va voir un moyen permettant de définir en intention des relations de préférences.

2. 1. 3. Combinaison de relations de préférence

Les relations de préférence peuvent être composées de plusieurs manières. On distingue généralement des compositions unidimensionnelles et multidimensionnelles:

- ***Composition unidimensionnelle*** : les relations de préférence composées sont définies sur un ensemble. Le résultat de cette composition est une nouvelle relation de préférence définie sur le même ensemble.
- ***Composition multidimensionnelle*** : à partir des relations de préférence définies sur des ensembles différents, on définit une nouvelle relation de préférence sur le produit cartésien de ces ensembles.

Ces compositions sont présentées dans les sections suivantes.

2. 1. 3. 1. Composition unidimensionnelle

Étant données deux relations de préférence $>_1$ et $>_2$ sur un ensemble d'éléments E, le résultat d'une composition unidimensionnelle de $>_1$ et $>_2$ est une autre relation $>_{12}$ sur E.

Des exemples de compositions sont la composition booléenne (union, intersection et différence), la fermeture transitive et la composition prioritaire.

***a) Composition booléenne* :** La relation $\succ_{12}$ est obtenue de façon classique en calculant l'union, l'intersection ou la différence des relations $\succ_1$ et $\succ_2$.

Par exemple, la composition par l'intersection est définie comme suit :

Définition 3.3 (- intersection). *Soient $\succ_1$ et $\succ_2$ deux relations sur E, la composition par intersection de $\succ_1$ et $\succ_2$, notée $\succ_{12} = \succ_1 \cap \succ_2$, est la relation de préférence définie par : pour tout x, y ∈ E, $x \succ_{12} y \equiv x \succ_1 y \wedge x \succ_2 y$*

b) Composition prioritaire : *C'est la composition de préférences prioritaires dites aussi préférences hiérarchiques.*

Définition 3.4 (- composition prioritaire). *Soient $\succ_1$ et $\succ_2$ deux relations sur E, la composition prioritaire de $\succ_1$ et $\succ_2$, notée $\succ_{12} = \succ_1 \lhd \succ_2$, est la relation de préférence définie par : pour tout x, y ∈ E, $x \succ_{12} y \equiv x \succ_1 y \vee (x \mathrel{/\!\!/_1} y \wedge x \succ_2 y)$ [Chomicki, 2003]*

La relation de préférence prioritaire $\succ_{12}$ obtenue par cette composition signifie que $\succ_1$ est plus importante que $\succ_2$.

Par rapports aux relations de préférence obtenues par composition unidimensionnelle présentées ci-dessus où les éléments comparés sont des constantes, les compositions suivantes permettent de passer à l'ordonnancement de tuples.

2. 1. 3. 2. Composition multidimensionnelle

Étant données deux relations de préférence $\succ_1$ sur E_1 et $\succ_2$ sur E_2, le résultat d'une composition multidimensionnelle de $\succ_1$ et $\succ_2$ est une relation de préférence $\succ_{12}$ sur $E_1 \times E_2$. La composition multidimensionnelle permet ainsi de définir des relations de préférence sur des ensembles de tuples à partir de relations de préférence sur des ensembles d'éléments atomiques.

Des exemples de compositions multidimensionnelles sont la composition *Pareto* et la composition *lexicographique.*

1. ***Composition Pareto :*** C'est la composition de préférences de la même importance.

Définition 3.5 (- composition Pareto)*. Soient deux relations de préférence $\succ_1$ sur E_1 et $\succ_2$ sur E_2, la composition Pareto de $\succ_1$ et $\succ_2$, notée $\succ_{12} = \succ_1 \otimes \succ_2$, est une relation de préférence définie par : pour tout $x = (x_1, x_2)$ et $y = (y_1, y_2)$ éléments de $E_1 \times E_2$,*

$x \succ_{12} y \equiv (x_1 \succ_1 y_1 \wedge (x_2 \succ_2 y_2 \vee x_2\ \theta_2\ y_2)) \vee (x_2 \succ_2 y_2 \wedge (x_1 \succ_1 y_1 \vee x_1\ \theta_1\ y_1))$

où $\theta_1 \in \{\|_1, =\}$, $\theta_2 \in \{\|_2, =\}$, c.-à-d. l'incomparabilité ([Chomicki, 2003]) ou l'égalité ([Kießling, 2002]).

La composition *Pareto* signifie que $\succ_1$ et $\succ_2$ sont également importantes.

2. ***Composition lexicographique*** **:** C'est la composition de préférences selon un ordre lexicographique.

Cette composition peut être définie comme suit.

Définition 3.6 (- composition lexicographique). *Soient deux relations de préférence* $\succ_1$ *sur* E_1 *et* $\succ_2$ *sur* E_2*, la composition lexicographique de* $\succ_1$ *et* $\succ_2$*, notée* $\succ_{12} = L(\succ_1, \succ_2)$*, est la relation de préférence définie par :*

pour tout $x = (x_1, x_2)$ *et* $y = (y_1, y_2)$ *éléments de* $E_1 \times E_2$, $x \succ_{12} y \equiv x_1 \succ_1 y_1 \vee$

$(x_1 \theta_1 y_1 \wedge x_2 \succ_2 y_2)$ *où* $\theta_1 \in \{//_1, =\}$.

En composant des relations de préférence, on souhaite que la nouvelle relation obtenue par composition soit elle aussi une relation de préférence. Ceci est toujours vérifié, cependant les propriétés des relations de préférence composantes ne sont pas toujours conservées par toutes les compositions comme le montre [Chomicki, 2003].

3. Modélisation de profil

Un profil utilisateur peut être composé de : **(1)** préférences sur le contenu de la réponse à la requête, **(2)** préférences sur la présentation de la réponse, et/ou **(3)** des conditions d'exploitation (les conditions matérielles et/ou les contraintes utilisateur). Nous présentons dans les paragraphes suivants le détail de ces composants.

***Le contenu* :** pour déterminer le contenu personnalisé, le profil doit porter sur les tuples ou les attributs de la réponse à la requête. Par exemple, [Agrawal and Wimmers 2000] offre un moyen d'exprimer les préférences du profil utilisateur à l'aide de fonctions affectant des scores aux tuples de la réponse et permettant de les ordonner selon leur importance et ainsi de déterminer quels sont les tuples les plus intéressants.

Notons qu'en *SQL*, l'ordonnancement des tuples peut être fait par l'intermédiaire de la clause *order by* qui permet de classer et de trier les résultats *SQL*, mais *order by* ne correspond pas à la prise en compte des préférences utilisateur. Et l'ordonnancement des attributs apparaît dans la clause select.

La présentation : afin de faciliter la tâche d'analyse de la masse de données, la personnalisation de la présentation permet de construire une façon adaptée pour afficher le contenu personnalisé. On fait intervenir ici des informations du profil sur la présentation telle que la disposition des informations à l'affichage. Ainsi, l'information affichée, même si c'est une réponse à une même requête, n'est pas identique pour tous les utilisateurs qui ont souvent des besoins différents.

Ce type de personnalisation n'a pas été beaucoup traité dans les *BDD*. [Das et al., 2006] ont introduit *la sélection et l'ordonnancement d'attributs* comme nouveau modèle d'extraction d'informations de *BDD* qui complète le modèle de ré-ordonnancement (ou de classification) de tuples de résultat d'une requête

[Agrawal et al., 2003] [Chaudhuri et al., 2004]. L'approche est basée sur l'importance de l'attribut. L'avantage de la méthode est de constituer une autre manière de présenter moins d'attributs et de faciliter l'exploration des tuples de la réponse selon leur importance :

- La sélection permet de déterminer les attributs utiles à afficher, et
- L'ordonnancement permet d'ordonner les attributs sélectionnés par ordre décroissant de leur utilité à l'affichage.

Les auteurs définissent plusieurs variantes pour la notion d'utilité d'attribut (le score, l'ordre, l'ordre relatif, etc.). Ensuite, ils proposent une approche qui retourne et affiche les *Top* attributs de chacune de ces variantes côte à côte.

Les conditions d'exploitation : on trouve dans cette partie les conditions liées au matériel utilisé par l'utilisateur et/ou les contraintes liées à l'utilisateur lui-même que nous allons distinguer en deux types dans ce qui suit.

Afin de considérer la diversité croissante des dispositifs d'accès à l'information, les systèmes de personnalisation proposés permettent de prendre en compte les caractéristiques techniques des dispositifs utilisés. La taille d'affichage, la capacité de la mémoire, la vitesse du réseau, etc., varient ainsi en fonction du matériel : *PC*, ordinateur portable, téléphone *WAP*, *PDA*, etc. La description des caractéristiques matérielles est utilisée dans le but d'afficher la réponse à la requête dans un format cohérent avec les capacités du dispositif de sortie. Ces caractéristiques sont vues comme des contraintes d'ordre matériel, appelées *contraintes techniques*, que l'on doit satisfaire et respecter lors de l'exécution de la requête et la présentation du résultat.

Par exemple [Koutrika and Ioannidis, 2005a] proposent de prendre en compte les contraintes suivantes :

– ***Cost*** : le coût d'exécution d'une requête doit être réduit au minimum ou doit satisfaire une certaine limite supérieure.

– ***Size*** : par définition, la personnalisation de requête vise d'une part à avoir de plus petites réponses. D'autre part, les réponses vides sont toujours indésirables. Par conséquent, la taille de la réponse doit être comprise dans un intervalle donné.

Un deuxième type de contraintes peut être lié à l'utilisateur lui même et qu'on peut appeler *contraintes utilisateur*. Ce sont des règles qui vont déterminer par exemple le type de la présentation du résultat souhaité par un utilisateur suivant sa fonction : préférer une présentation sous forme d'un graphe à une présentation sous forme d'un arbre.

3.1. Modèle de profil proposé par Koutrika et Ioannidis

Le langage de préférence proposé par [Koutrika and Ioannidis, 2004] permet de représenter des préférences quantitatives. Les préférences utilisateur sont définies en utilisant des fonctions de *score* qui associent des degrés d'intérêt aux requêtes.

En particulier, le langage de préférence affecte des degrés d'intérêt sur les conditions de sélection d'une requête. Ainsi, une requête est plus intéressante qu'une autre si son degré d'intérêt est plus élevé. Les préférences sont dites atomiques et sont de la forme <q, d> avec q la condition atomique et le degré d'intérêt d ∈ [0, 1].

Exemple 3.3. Soit une relation *DIRECTOR (did, name)* d'une base de données *"movies"*. La préférence atomique *‹ DIRECTOR.name = "W.Allen", 0.9 ›* exprime un intérêt avec un degré 0.9 pour le réalisateur " W. Allen". Cet intérêt est exprimé sur la condition *DIRECTOR.name="W. Allen".*

3. 1. 1. Types de préférences

a) Préférences de sélection (atomiques)

Le profil de l'utilisateur contient des données qui expriment ses habitudes, des prédicats fréquemment utilisés dans ses requêtes ou des définitions de l'ordre de préférences de ces prédicats [Koutrika et Ioannidis 2004]. L'intérêt de l'utilisateur pour chacun de ces éléments est exprimé par un *degré* qui est un nombre réel compris entre 0 et 1. Prenons par exemple une base de données dont le schéma est le suivant :

TRANSPORT (idT, moyen, Type_trajet, Confort)
HOTEL (idH, nombre_étoiles, région)
VOYAGE (idV, prix, lieu_départ, lieu_arrivée, nombre_jours, idH, idT)
DEPART (idD, idV, date, heure)

Si un utilisateur qui habite à *Paris* et qui aime voyager pendant le week-end, descend d'habitude dans des hôtels au *centre ville* et préfère voyager en *train* plutôt qu'en *car*, son profil peut être écrit sous la forme :

Exemple 3.4 de profil :

{ TRANSPORT.idT = VOYAGE.idT	*1*	*(a)*
HOTEL.idH = VOYAGE.idH	*1*	*(b)*
VOYAGE.lieu_départ = 'Paris'	*1*	*(c)*
HOTEL.région = 'centre ville'	*0.9*	*(d)*
VOYAGE.nombre_jours = 2	*0.7*	*(e)*
TRANSPORT.moyen = 'train'	*0.7*	*(f)*
TRANSPORT.moyen = 'car'	*0.5 }*	*(g)*

Sur chaque expression du profil, considérée comme une sous requête, est ajouté un nombre compris entre 0 et 1 pour exprimer l'importance relative de cette expression par rapport aux autres. Ainsi la valeur 1 sur les trois premières expressions signifie que ces conditions doivent être toujours satisfaites.

Les autres expressions expriment le fait que l'utilisateur a une plus forte préférence pour les hôtels situés au centre ville (d) que pour les voyages de deux jours (e) et qu'il préfère voyager en train (f) plutôt qu'en car (g).

***b) Préférences de jointure* :** une préférence d'un utilisateur pour une condition de jointure q est 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. De plus, les préférences expriment une dépendance entre la partie gauche de la jointure, qui exprime que la relation est toujours incluse dans la requête, et la partie droite correspond à la relation qui peut inclure une influence par le résultat final si la jointure a été prise en compte.

c) Les préférences implicites

Les préférences peuvent être implicites et elles sont celles dérivées des préférences utilisateur atomiques. Le formalisme permettant les compositions de préférences utilisé dans les travaux de Koutrika et Ioannidis est le graphe des préférences et est décrit ci-après :

Représentation graphique des préférences utilisateur

Les préférences de l'utilisateur peuvent également être présentées sous forme d'un graphe où les nœuds représentent soit des relations, soit des attributs, soit des valeurs, et les arcs représentent soit des sélections (entre un nœud d'attribut et un nœud de valeur) soit des jointures (entre deux nœuds d'attribut). Les poids relatifs des prédicats sont représentés sur les arcs correspondants (Fig 3.2.).

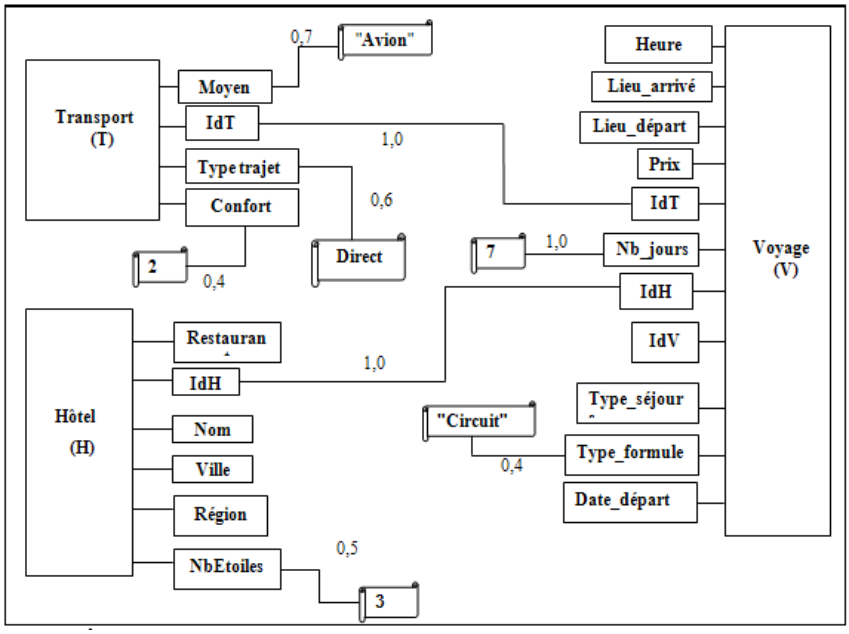

Fig 3.2. – Représentation graphique d'un profil utilisateur [Koutrika et Ioannidis 2004]

Exemple 3.5 **:** Préférences utilisateur

P1 =	
{ VOYAGE.idT=TRANSPORT.idT	1.0 (a)
VOYAGE.idH=HOTEL.idH	1.0 (b)
VOYAGE.nbre_jours >7	1.0 (c)
TRANSPORT.moyen='avion'	0.7 (d)
TRANSPORT.type_trajet='direct'	0.6 (e)
HOTEL.nbre_etoiles > 3	0.5 (f)
VOYAGE.type_form ule<>'circuit'	0.4 (g)
TRANSPORT.confort>2	0.4 (h)}

Ce formalisme permet les compositions de préférences suivantes :

- En composant les préférences utilisateur atomiques qui sont adjacentes dans le graphe, on peut construire des préférences transitives (ou implicites),
- Etant donné un ensemble de préférences utilisateur, atomiques ou transitives, on peut former des combinaisons logiques pour obtenir des préférences complexes conjonctives et disjonctives.

Dans le premier cas, les préférences transitives définies ci-dessous, sont exprimées à travers les liens entre les relations dans le graphe.

3. 1. 2. Préférence transitive

Définition 3.7 (- préférence transitive). *Soient $P_N = \{P_1, \ldots, P_N\}$ un ensemble de N préférences atomiques composables et $D_N = \{d_i \mid d_i$: degré d'intérêt dans $P_i \in P_N$, $i \in [1,N]\}$ l'ensemble de degrés d'intérêt correspondants. La fonction de préférence transitive $f_\otimes$ est définie par : $f_\otimes (D_N)) = d_1 \times d_2 \times \ldots \times d_N$*

Une préférence transitive est composée des préférences atomiques de P_N. On note que chaque fonction $f_\otimes$ calculant le degré d'intérêt dans cette préférence doit vérifier la condition : $f_\otimes(D_N) \leq \min(D_N)$.

C'est-à-dire, plus la longueur du chemin dans une préférence transitive augmente plus le degré d'intérêt correspondant diminue. Ainsi, la multiplication est choisie par les auteurs comme fonction $f_\otimes$.

Dans le deuxième cas, étant donné un ensemble de préférences utilisateur, atomique ou transitif, on peut former des combinaisons logiques de ces préférences pour obtenir des préférences conjonctives ou disjonctives complexes.

3. 1. 3. Composition de préférences

Soient P_N un ensemble de *N* préférences (atomiques ou transitives) et

$D_N = \{d_i \mid d_i$: degré d'intérêt dans $P_i \in P_N, i \in [1, N]\}$ l'ensemble de degrés d'intérêt correspondants.

Définition 3.8 (- préférence conjonctive). *La préférence conjonctive est définie par :*

$f_\wedge(D_N) = 1 - (1 - d_1)(1 - d_2) \ldots (1 - d_N) = 1 - \Pi_{i=1}^{N}(1 - d_i)$

Chaque fonction $f_\wedge$ calculant le degré d'intérêt dans cette préférence doit vérifier la condition : $f_\wedge(D_N) \geq max(D_N)$.

Définition 3.9 (- préférence disjonctive). *La préférence disjonctive est définie par :*

$f_\vee(D_N) = (d_1 + d_2 + \ldots + d_N)/N$

Chaque fonction $f_\vee$ calculant le degré d'intérêt dans cette préférence doit vérifier la condition : $min(D_N) \leq f_\vee(D_N) \leq max(D_N)$

Le choix de ces deux fonctions est justifié par l'intuition que les plus petites réponses à une requête sont d'un intérêt plus élevé pour un utilisateur.

Exemple 3.6 Soient deux relations MOVIE (mid, title, year) et DIRECTOR(did, name) d'une base de données « movies ».

< DIRECTOR.name = "W.Allen", 0.9> *Préférences atomiques stockées*
< MOVIE.mid = DIRECTOR.did, 0.8>

< MOVIE.mid = DIRECTOR.did and DIRECTOR.name = ″W.Allen″, 0.9 × 0.8> *Préférences implicites*

Dans [Koutrika and Ioannidis, 2004] seules les préférences dites *positives* et *exactes* de présence sont prises en compte. Ces préférences sont du type : *'I like actor W. Allen'*.

Ce formalisme est généralisé dans [Koutrika and Ioannidis, 2005b] permettant de formuler des préférences qui ne sont pas considérées par [Koutrika and Ioannidis, 2004] dans lequel le nouveau type de préférence considéré est :

- Préférence **variable** : *'I like films with duration around 2h'*,
- Préférence **négative** : *'I do not like thrillers'* ou
- Concernant l'absence des valeurs : *'I like movies without violence'*.

Ces opérateurs seront détaillés dans les paragraphes suivants.

3. 2. Modèle de profil proposé par Kießling

Le langage proposé par [Kießling 2002] adopte l'approche qualitative et quantitative. Il propose un langage formel pour la formulation de relations de préférence et de fonctions de score.

Les préférences utilisateur portent sur des domaines d'attributs et des tuples d'une relation de base de données.

Dans le formalisme de Kießling, toute expression de préférence est définie à partir de constructeurs de préférence dits de base. Ces constructeurs de base permettent de définir en intention des relations de préférence sur le domaine d'un attribut. Ils sont de deux types suivant que les valeurs des domaines sont de type numérique ou nominal.

3. 2. 1. L'approche quantitative

L'expression des préférences est basée sur l'attribution de poids aux attributs ou aux tuples de la BDD en utilisant une fonction d'utilité. La définition d'une fonction d'utilité dans ce formalisme est basée sur la définition d'un constructeur de préférence de base, appelé *SCORE*, présentée ci-après.

Définition 3.10 (- Préférence SCORE). Soit l'attribut A de domaine dom(A). Étant donnée une fonction d'utilité f : dom(A) → R, un constructeur de préférence, noté P := SCORE(A, f), définit la relation de préférence $\succ_P$ par :

pour tout x, y ∈ dom(A), $x \succ_P y \equiv f(y) < f(x)$ où < est l'ordre ″est inférieur à″ sur R.

Composition des préférences

L'opérateur $rank_F$ défini ci-dessous (voir définition 3.11), est l'unique opérateur basé sur une fonction de score (voir définition 3.10). Les différents scores sont accumulés en un seul score global par l'application d'une fonction de combinaison monotone multi-attributs, notée *F*.

Définition 3.11 (- expression de préférence numérique). Étant données les expressions de préférence P_1 = SCORE(A_1, f_1) et P_2 = SCORE(A_2, f_2) et une fonction de combinaison F : R × R → R, l'expression de préférence P = $rank_F(P_1, P_2)$, appelée expression de préférence numérique, définit la relation de préférence $\succ_P$ par :

pour tout x = (x_1, x_2) et y = (y_1, y_2) éléments de dom(A_1) × dom(A_2),

x $\succ_P$ y ≡ F($f_1(y_1)$, $f_2(y_2)$) < F($f_1(x_1)$, $f_2(x_2)$)

$rank_F$ est un constructeur de préférence qui peut être appliqué uniquement avec *SCORE*.

3. 2. 2. L'approche qualitative

a) Préférences sur des attributs à valeurs non numériques

- ***Préférence POS***. Soit un ensemble fini de valeurs préférées d'un attribut A de domaine dom(A), noté POS-set ⊆ dom(A). Un constructeur de préférence positive, noté : *P* := *POS*(*A*, POS-set), définit la relation de préférence $\succ_p$ par :

pour tout x, y ∈ *dom*(*A*), $x >_P y \Leftrightarrow y \notin$ POS-set ∧ $x \in$ POS-set.

On dit que *x* est préférée à *y* si et seulement si *x* du domaine de l'attribut *A* appartient à l'ensemble de valeurs *POS-set* et *y* non.

Exemple 3.7 On suppose qu'un utilisateur exprime ses préférences parmi les valeurs de l'attribut ″Color″ en spécifiant sa préférence pour la couleur rouge.

---

s_1 *préférer une voiture de couleur rouge.*

On peut exprimer la préférence s_1 par le constructeur de préférence P := POS(Color, {red}) qui signifie que ″red″ est préférée à toutes les autres couleurs du domaine dom(Color) = {black, white, yellow, silver, red} de l'attribut ″Color″.

La relation préférence, notée $\succ_p$, peut être définie en utilisant la formule de préférence suivante :

pour x, y ∈ dom(Color), y $\succ_p$ x ≡ (y = ″red ″) ∧ (x = ″black″ ∨ x = ″white″ ∨ x = ″yellow ″ ∨ x = ″silver″)

La comparaison entre les couleurs : ″black″, ″white″, ″yellow″ et ″silver″ n'est pas possible vis à vis de la préférence s_1. Mais, toutes ces valeurs sont considérées comme des alternatives ou des solutions possibles si la couleur ″red″ n'est pas disponible.

- ***Préférence NEG*** Comme la préférence positive, la préférence négative NEG est définie par rapport à un ensemble de valeurs NEG-set donné en paramètre. La seule

différence est le fait que cette fois les éléments préférés sont ceux dont la valeur de l'attribut A n'appartient pas à cet ensemble : P est une préférence négative i.e.

P := NEG (A, NEG -set), définit la relation de préférence $\succ_P$ par :

pour tout x, y ∈ dom(A), $x >_P y \Leftrightarrow x \notin$ NEG -set ∧ y ∈ NEG -set.

b) Préférences sur des attributs à valeurs numériques

Un deuxième type de préférences décrit dans par [Kießling 05] sont les préférences numériques qui sont le plus souvent associées à la comparaison de valeurs numériques. En présence d'opérateurs calculant la similarité entre les éléments, il est possible d'appliquer ces opérateurs aussi à des valeurs textuelles. Les préférences présentées sont : *autour de, entre, plus petit* et *plus grand*.

- ***La préférence autour de (AROUND)*** exprime l'intérêt de l'utilisateur vers les éléments dont la valeur de l'attribut A est la plus proche possible de la valeur donnée en paramètre. Pour le calcul de la distance, les auteurs de l'article prennent une fonction de similarité, ce qui permet d'utiliser la préférence *″autour de″* pour des chaînes de caractères. De façon formelle cette préférence est définit par :

Soit z une valeur du domaine de A. P = AROUND (A, z).

$x <_P y$ ssi Similarité (y, z) > Similarité (x, z).

En d'autres termes on choisit les éléments dont la valeur pour l'attribut A est la plus proche de z.

- ***La préférence « entre » (BETWEEN)*** comme son nom l'indique permet à l'utilisateur d'exprimer ses préférences pour les éléments dont la valeur pour un attribut donné A est comprise dans un intervalle :

Soient *max* et *min* deux valeurs du domaine de A telles que *max* > *min*. Dans ce cas

P = BETWEEN (A, [min, max])

$x <_P y$ ssi distance(x, [min, max]) < distance(y, [min, max]) où la fonction de distance est définie comme suit :

distance(v, [min, max]) = 0 si v∈[min, max] ; (min – v) si v<min et (max –v) si v>max

Les préférences plus petit (LOWEST) et plus grand (HIGHEST) cherchent la plus petite ou la plus grande valeur d'un attribut d'un ensemble d'éléments.

- ***Préférence LOWEST***. Soit l'attribut A de domaine dom(A). Un constructeur de préférence plus petit, noté P := LOWEST(A), définit la relation de préférence $\succ_P$ par :

pour tout x, y ∈ dom(A), $x \succ_P y \Leftrightarrow y > x$

où > est le symbole mathématique ″est supérieur à″.

On dit que x est préférée à y si et seulement si y du domaine de l'attribut A est supérieure à x du même domaine.

Exemple 8 Soit la préférence utilisateur suivante :

---

s_2 préférer une voiture plus ancienne.

On peut exprimer la préférence s_2 par le constructeur de préférence P := LOWEST(Year) qui signifie que l'utilisateur préfère des voitures anciennes (c.-à-d. des voitures des années passées).

Dans ce sens, la relation préférence, notée $\succ_P$, peut être définie en utilisant la formule de préférence suivante : pour x, y $\in$ dom(Year), x $\succ_P$ y $\equiv$ y > x

De la même façon la préférence HIGHEST cherche la plus grande valeur d'un attribut d'un ensemble d'éléments.

c) Composition des expressions de préférence

Des préférences dites *complexes*, définissent en intention des relations de préférence sur des domaines d'attributs (dans le cas de composition unidimensionnelle) ou sur des tuples de relation (dans le cas de composition multidimensionnelle) peuvent êtres exprimées en utilisant des compositions sur des préférences simples.

Les compositions booléennes décrites sont l'intersection, l'union disjointe (définition 3.12) et la somme linéaire (définition 3.13) pour le cas unidimensionnel.

Soient deux constructeurs de préférence de base P_1 et P_2 définissant respectivement des relations de préférence $\succ_{P1}$ et $\succ_{P2}$ sur des attributs A_1 et A_2 de domaines disjoints, c.-à-d. dom(A_1)$\cap$dom(A_2) = $\emptyset$. Soit l'attribut A de domaine dom(A_1)$\cup$dom(A_2).

Définition 3.12 (- Union disjointe) *L'expression de préférence P_1 + P_2, appelée union disjointe de P_1 et P_2, définit la relation de préférence $\succ_{P_1+P_2}$ par :*

$$x \succ_{P_1+P_2} y \equiv x \succ_{P_1} y \vee x \succ_{P_2} y$$

Définition 3.13 (- Somme linéaire) *L'expression de préférence $P_1 \oplus P_2$, appelée somme linéaire de P_1 et P_2, définit la relation de préférence $\succ_{P_1 \oplus P_2}$ par :*

$$x \succ_{P_1 \oplus P_2} y \equiv x \succ_{P_1} y \vee x \succ_{P_2} y \vee (x \in dom(A_1) \wedge y \in dom(A_2))$$

Pour le cas multidimensionnel, on trouve en plus des compositions *Pareto* et *lexicographique.*

3. 3. Modèle de profil proposé par Chomicki

Chomicki propose une approche formelle qualitative de formulation de préférences et leur intégration dans les langages de l'algèbre relationnelle.

Les préférences utilisateur portent sur des *tuples* d'une base de données. Chaque paire de tuples de la relation de préférence qualitative indique la préférence d'un tuple par rapport à un autre tuple.

Le formalisme d'expression des préférences décrit est basé sur des relations binaires de préférence, c.-à-d. que toute relation binaire est considérée comme une relation de préférence.

Les relations binaires de préférence sont définies à travers des formules de préférence. Ces formules de préférence sont définies comme suit. Soit R une relation de schéma $R(\mathrm{A})$ où $\mathrm{A} = \{A_1, \ldots, A_k\}$ est un ensemble d'attributs avec :

$dom(\mathrm{A}) = dom(A_1) \times \ldots \times dom(A_k)$.

Définition 3.14 (- formule de préférence) *Étant donnés deux k-uplets* $t_1, t_2 \in dom(\mathrm{A})$*, une formule de préférence, notée* $C(t_1, t_2)$*, est une formule de premier-ordre définissant une relation de préférence ,* $\succ_C$ *comme suit :* $t_1 \succ_C t_2 \equiv C(t_1, t_2)$

Exemple 3.9 On suppose qu'un utilisateur exprime les préférences suivantes sur les tuples de la relation ″Car″ de l'exemple 3.1 :

---

s_3 préférer les voitures de modèle 'Escort' par rapport à celles de modèle 'Corolla'.

Pour s_3, on définit une relation de préférence notée, $\succ_C$, en utilisant une formule de préférence C par :

$$(mk, m, c, p, y) \succ_C (mk', m', c', p', y') \equiv C((mk, m, c, p, y), (mk', m', c', p', y'))$$
$$\equiv m = \text{'Escort'} \wedge m' = \text{'Corolla'}$$

qui traduit le fait que des voitures de modèle *″Escort″* sont préférées à celles de modèle *″Corolla″*.

3. 3. 1. Combinaison des préférences

Dans certains cas, il est possible que l'utilisateur ait besoin d'exprimer de nouvelles préférences plus complexes. Celles-ci peuvent être obtenues par des combinaisons de relations de préférence afin de répondre à la requête de l'utilisateur. [Chomicki, 2003] étudie cinq types de composition entre des relations de préférence : composition booléenne, fermeture transitive, composition par priorités, composition Pareto et composition lexicographique.

3. 3. 2. Préférences conditionnelles

Ce formalisme permet aussi à l'utilisateur d'exprimer des préférences de type conditionnel. Il offre un moyen d'exprimer les dépendances entre les éléments sur lesquels portent les préférences utilisateur. Une manière de faire est d'inclure une formulation particulière de conditions dans l'expression de la formule de préférence. L'exemple ci-dessous illustre les préférences de ce type.

Exemple 3.10 Soient deux relations Wine(Name, Type) et Dish(Name, Type) et une vue ″Meal″ qui contient les compositions possibles d'un repas.

```
CREATE VIEW Meal (Dish, DishType, Wine, WineType) AS SELECT * FROM Wine, Dish;
```

Soit l'expression de préférence suivante sur les tuples de ″Meal″.

---

s_4 dans un repas, préférer le vin blanc en présence des poissons et le vin rouge en présence de viande.

Nous exprimons ces préférences par une relation de préférence, notée $\succ_{C1}$, en utilisant une formule C_1 comme suit :

$$(d, dt, w, wt) \succ_{C1} (d', dt', w', wt') \equiv (d = d' \wedge dt = \text{'fish'} \wedge wt = \text{'white'} \wedge dt' = \text{'fish'} \wedge wt' = \text{'red'}) \vee (d = d' \wedge dt = \text{'meat'} \wedge wt = \text{'red'} \wedge dt' = \text{'meat'} \wedge wt' = \text{'white'})$$

Notons que ceci forcera tout vin blanc à être préféré à tout vin rouge pour un repas avec du poisson, et tout vin rouge à être préféré à tout vin blanc pour un repas avec de la viande. Pour les autres genres de plats, aucune préférence n'est indiquée. Ces préférences sont donc conditionnelles, puisqu'elles dépendent du type du plat considéré.

3. 3. 3. Préférences de type Ceteris Paribus

Le formalisme de Chomicki permet à l'utilisateur d'exprimer des préférences de type *Ceteris Paribus* car avec les formules de préférence, on peut formuler des préférences sur plusieurs attributs à la fois.

Exemple 3.11 Soient les préférences utilisateur suivantes sur les tuples de la relation ″Car″ de l'exemple 3.1 :

---

s_5 préférer les BMW noires aux Ferrari rouges.

Pour s_5, on définit une relation de préférence notée, $\succ_{C2}$, en utilisant une formule de préférence C_2 par :

$$(mk, m, c, p, y) \succ_{C2} (mk', m', c', p', y') \equiv C_2((mk, m, c, p, y), (mk', m', c', p', y'))$$
$$\equiv mk = \text{'BMW'} \wedge c = \text{'black'} \wedge mk' = \text{'Ferrari'} \wedge c' = \text{'red'} \wedge m = m' \wedge p = p' \wedge y = y'$$

Dans cet exemple, on exprime des préférences pour des ″BMW noires″ par rapport aux ″Ferrari rouges″ toutes chose étant égales par ailleurs, c.-à-d. les valeurs des autres attributs : ″model″, ″price″ et ″year″ doivent être identiques. En particulier, dans notre exemple, on va avoir $t_7 \succ_{C2} t_6$.

4. La personnalisation des requêtes

La personnalisation dans le domaine des bases de données a été envisagée sous deux aspects :

(i) L'utilisation du profil de l'utilisateur pour la reformulation de la requête.

(ii) L'utilisation des langages d'expression des requêtes.

4. 1. La personnalisation à l'aide des profils

4. 1. 1. Techniques de reformulation de requêtes

Exploiter le profil de l'utilisateur pour reformuler sa requête en y intégrant des éléments de son centre d'intérêt ou de ses préférences. Cette technique d'enrichissement, courante dans les langages à mots clés en recherche d'information, est très récente en bases de données. Elle est utilisée par Koutrika Et Ioannidis [Koutrika et Ioannidis 2004], [Koutrika et Ioannidis 2005a].

On rappelle que, Koutrika adopte l'approche quantitative pour l'expression des préférences, le profil de l'utilisateur est composé d'un ensemble de prédicats pondérés. Le poids d'un prédicat exprime son intérêt relatif pour l'utilisateur.

Ces préférences définissent un ordre total sur des requêtes, et sont stockées dans des profils utilisateurs. Dans ce cas, deux requêtes sont comparées sans accéder aux données. Une requête est plus intéressante qu'une autre si son degré d'intérêt est plus élevé.

La requête posée par l'utilisateur est transformée selon ce profil suivant le processus décrit ci-après.

4. 1. 2. Processus d'enrichissement

Prenons par exemple une base de données dont le schéma est le suivant :

Exemple 3.12 : Exemple de schéma d'une base de données

TRANSPORT (idT, moyen, direct, niveauConfort)
HOTEL (idH, nom, nbEtoiles, region, ville)
VOYAGE (idV, prix, lieuDep, lieuArr, nbJours, typeSejour, idH, idT, idD)
DEPART (idD, date, heure, pointRDV)

Et soit un utilisateur ayant les préférences suivantes :

- il préfère partir de *Toulouse* mais en cas de besoin, peut facilement aller à *Paris*,
- il aime les *circuits touristiques* de *plus de 7 jours*,
- il descend d'habitude dans des hôtels d'*au moins 3 étoiles* au *centre ville*,
- il ne veut pas dépenser *plus de 1000* euro,
- il préfère voyager en *avion* plutôt qu'en *train*,
- il aime les voyages *directs*, avec un *niveau de confort d'au moins 2 étoiles.*

Ces préférences peuvent être exprimées avec l'ensemble de prédicats suivants :

Exemple 3.13 : Exemple de profil utilisateur composé de prédicats

Profil P1 : { VOYAGE.idT = TRANSPORT.idT	*1.0*	*(a)*
VOYAGE.idD = DEPART.idD	*1.0*	*(b)*
VOYAGE.idH= HOTEL.idH	*0.8*	*(c)*
HOTEL.idH = VOYAGE.idH	*0.5*	*(d)*
VOYAGE.nbJours > 7	*1.0*	*(e)*
VOYAGE.lieuDep = 'Toulouse'	*0,8*	*(f)*
TRANSPORT.moyen='avion'	*0.7*	*(g)*
DEPART.heure > 22hOO	*0.65*	*(h)*
TRANSPORT.direct=VRAI	*0.6*	*(i)*
VOYAGE.prix < 1000	*0.55*	*(j)*
HOTEL.nbEtoiles > 3	*0.5*	*(k)*
VOYAGE.typeSejour = 'circuit'	*0.5*	*(l)*
TRANSPORT.niveauConfort > 2	*0.35*	*(m)*
VOYAGE.lieuDep = 'Paris'	*0.3*	*(n)*
HOTEL.region = 'centre ville'	*0.2*	*(o)*
TRANSPORT.moyen = 'car'	*0.1*	*(p)*

Le profil P_1 contient deux types de prédicats : (i) prédicats de sélection et (ii) prédicats de jointure. Le poids de chaque prédicat de sélection *q*, noté par *doi(q)*, exprime son importance par rapport aux autres prédicats de sélection. Par exemple, l'utilisateur a une plus forte préférence pour les voyages au départ de Toulouse (f) que pour les voyages au départ de Paris (n) et il préfère également voyager en avion (g) plutôt qu'en car (p). Les prédicats de sélection ayant un poids égal à 1.0 (par exemple le prédicat (e)) sont considérés comme étant obligatoires et doivent être toujours satisfaits. À la différence des prédicats de sélection, les prédicats de jointure sont orientés. La partie gauche d'un prédicat de jointure désigne la relation qui doit être dans la requête pour que la relation de sa partie droite puisse y être ajoutée. Son poids exprime l'intérêt de l'utilisateur que les prédicats exprimés sur la relation de droite soient ajoutés à la requête utilisateur si cette requête contient la relation de gauche.

Par exemple, si la requête contient la relation *VOYAGE*, il est plus intéressant pour l'utilisateur de prendre en compte les prédicats exprimés sur *TRANSPORT* (a) que ceux exprimés sur *HOTEL* (c). De même, il est plus important de tenir compte des prédicats exprimés sur la relation *HOTEL* si la requête contient la relation *VOYAGE* (c) qu'inversement (d).

Le profil de l'utilisateur peut être présenté également sous forme d'un graphe :

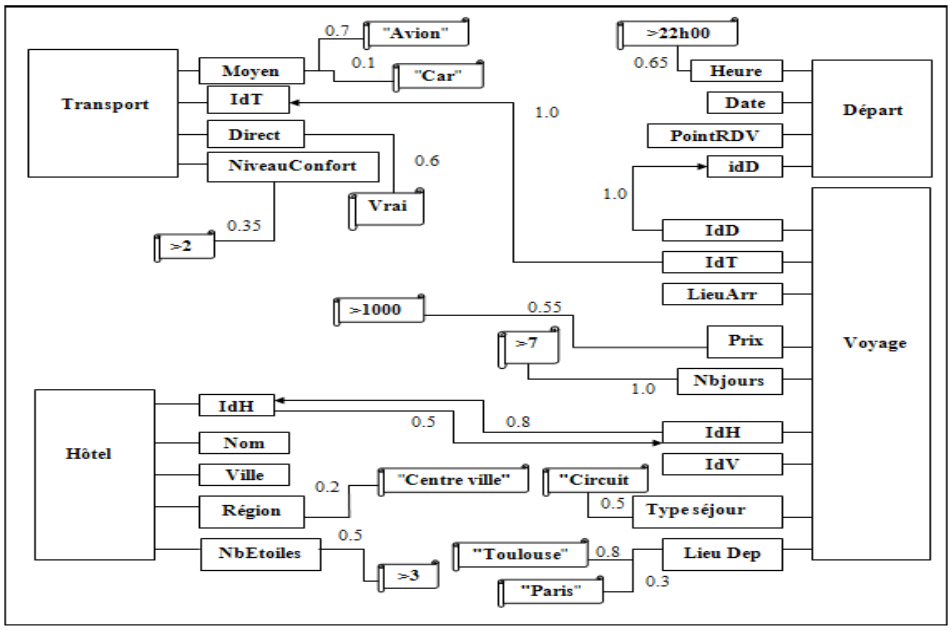

Fig. 3.3 – Représentation graphique du profil utilisateur

Les prédicats du profil utilisateur sont utilisés pour enrichir sa requête. Le processus d'enrichissement comporte deux phases principales :

- Sélection des prédicats qui vont enrichir la requête et
- Intégration de ces prédicats à la requête.

La suite de cette section, résume les méthodes proposées dans les travaux de Koutrika et Ioannidis pour traiter ces deux étapes.

a. Sélection des prédicats pertinents

La sélection des prédicats pour l'enrichissement de la requête utilisateur consiste à choisir les Top K prédicats qui sont en relations avec la requête et qui ne sont pas conflictuels avec elle [Koutrika and Ioannidis, 2004].

Un prédicat du profil utilisateur est en conflit avec la requête, s'il est en conflit avec un prédicat déjà présent dans la requête. Autrement dit, un prédicat du profil est conflictuel avec la requête si la conjonction de ce prédicat et ceux de la requête donne toujours un résultat nul.

Prenons par exemple le profil P_1 (Exemple 3.13) et une requête initiale Q_1 qui recherche des voyages de 3 jours à destination de ″Madrid″ dont la date de départ est le 10 novembre 2007 (Exemple 3.14).

***Exemple 3.14* :** Requête initiale Q_1

```
SELECT V.idV
FROM VOYAGE V, DEPART D
WHERE V.idD = D.idD AND D.date = '10/11/2007' AND
V.lieuArr = 'Madrid' AND V.nbJours = 3 ;
```

Le prédicat *« VOYAGE.nbJours > 7 »* de P_1 est conflictuel avec Q_1 parce que Q_1 contient déjà le prédicat *« VOYAGE.nbJours = 3 »* et les deux prédicats ne peuvent pas être satisfaits simultanément.

Rechercher les prédicats qui sont en relations avec la requête revient à rechercher les chemins dans le graphe représentant le profil utilisateur qui partent d'un nœud relation déjà présent dans la requête et qui se terminent par un nœud valeur. La Figure 3.4 montre un exemple de tel chemin pour la requête Q_1 de l'Exemple 3.14 et le profil utilisateur P_1 de l'Exemple 3.13. Sur cette figure, les nœuds représentant des éléments de la requête initiale ont un motif hachuré et les nœuds du chemin un motif en points. Pour pouvoir lier le prédicat *« HOTEL.nbEtoiles > 3 »* à Q_1, on doit utiliser le prédicat de jointure « *VOYAGE.idH = HOTEL.idH* ». La conjonction de ces deux prédicats est appelée préférence implicite.

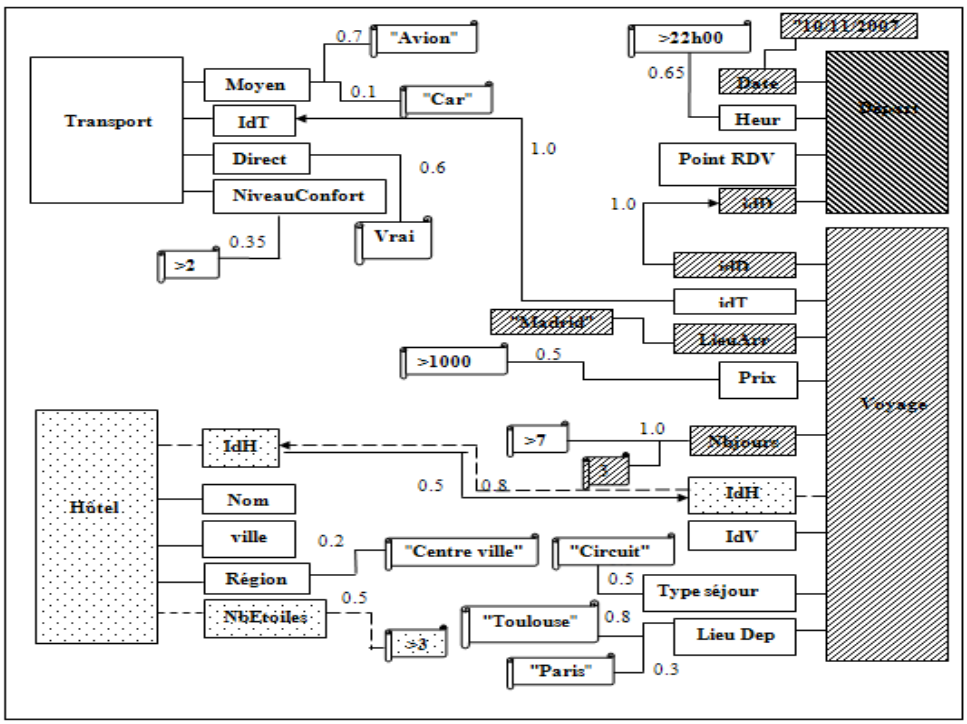

Fig. 3.4 – Exemple de prédicats du profil utilisateur P_1 lié à la requête Q_1

L'intérêt d'une préférence implicite est fonction des intérêts des prédicats la composant.

La règle est que l'intérêt d'une préférence décroît avec le nombre de prédicats qui la composent. La fonction choisie dans [Koutrika and Ioannidis, 2004] est la multiplication des degrés d'intérêt. En utilisant cette formule, l'intérêt de la préférence implicite de notre exemple est :

*doi (VOYAGE.idH= HOTEL.idH ∧ HOTEL.nbEtoiles > 3) = 0.8 * 0.5 = 0.4*

Suivant ce procédé, on peut déduire les autres préférences de P_1 qui sont liées à Q_1 et qui ne sont pas conflictuelles avec elle (Exemple 3.15).

***Exemple 3.15* :** Prédicats de P_1 pouvant enrichir Q_1

VOYAGE.lieuDep = 'Toulouse'	0,8 (i)
(VOYAGE.idT = TRANSPORT.idT ∧ TRANSPORT.moyen='avion')	0.7 (ii)
DEPART.heure > 22hOO	0.65 (iii)
(VOYAGE.idT = TRANSPORT.idT ∧ TRANSPORT.direct=VRAI)	0.6 (iv)
VOYAGE.prix < 1000	0.55 (v)
VOYAGE.typeSejour = 'circuit'	0.5 (vi)
(VOYAGE.idH= HOTEL.idH ∧ HOTEL.nbEtoiles > 3)	0.4 (vii)
(VOYAGE.idT = TRANSPORT.idT ∧ TRANSPORT.niveauConfort > 2)	0.35(viii)
VOYAGE.lieuDep = 'Paris'	0.3 (ix)
(VOYAGE.idH= HOTEL.idH ∧ HOTEL.region = 'centre ville')	0.16 (x)
(VOYAGE.idT = TRANSPORT.idT ∧ TRANSPORT.moyen = 'car')	0.1 (xi)

b. Intégration des prédicats pertinents à la requête

L'intégration des prédicats du profil à la requête est guidée par trois paramètres :

- ***Top K*** : nombre de prédicats du profil devant être pris en compte. La notion de Top K peut être exprimée de différentes manières : les K prédicats de plus grand poids, les prédicats dont le poids est supérieur à un seuil donné etc. Dans notre exemple, nous considérons les K prédicats de plus grand poids. On remarque que seuls les prédicats de sélection non contradictoires avec la requête sont pris en compte. Des prédicats de jointure sont ajoutés au cas où il faut ajouter une nouvelle relation à la requête.
- ***M*** : nombre de prédicats parmi les Top K qui doivent *obligatoirement* être satisfaits; ça correspond aux M prédicats de plus grand poids parmi les top K.
- **L** : nombre minimal de prédicats parmi les Top K-M restants que chaque tuple du résultat doit satisfaire.

Les grandes lignes du processus d'enrichissement de la requête utilisateur peuvent être résumées ainsi :

1. Choisir les Top K prédicats,

2. Ajouter les M prédicats de plus grand poids comme une conjonction aux prédicats de la requête utilisateur,

3. Calcul de tous les ensembles possibles de L prédicats parmi les K-M restants dont la conjonction n'est pas contradictoire. On appellera L-combinaison un ensemble de L prédicats parmi les K-M.

4. Ajouter la disjonction des conjonctions de L prédicats à la requête utilisateur.

Pour illustrer ce processus d'enrichissement des requêtes, considérons la requête Q_1 de l'Exemple 3.14 et fixons les paramètres K, L et M respectivement à 5, 2 et 2 sur l'ensemble de prédicats de P_1 pouvant enrichir Q_1 (Exemple 3.15). Le premier pas de l'algorithme va sélectionner les 5 prédicats de plus grand poids (i, ii, iii, iv et v). Les deux premiers prédicats sont ajoutés comme une conjonction à Q_1 qui devient :

***Exemple 3.16* :** Requête initiale Q_1 enrichie avec les prédicats obligatoires

```
SELECT V.idV
FROM VOYAGE V, DEPART D, TRANSPORT T
WHERE V.idD = D.idD AND D.date = '10/11/2007' AND
V.lieuArr = 'Madrid' AND V.nbJours = 3 AND
V.idT=T.idT AND V.lieuDep = 'Toulouse' AND T.moyen = avion';
```

Les combinaisons de 2 prédicats parmi les 3 qui restent sont :

a) DEPART.heure>22h00 ∧ (VOYAGE.idT = TRANSPORT.idT ∧ TRANSPORT.direct=VRAI)

b) DEPART.heure>22h00 ∧ VOYAGE.prix < 1000

c) (VOYAGE.idT = TRANSPORT.idT ∧ TRANSPORT.direct=VRAI) ∧ VOYAGE.prix < 1000

Finalement, Exemple 3.17 et Exemple 3.18 représentent respectivement le résultat final de l'enrichissement de Q_1 sous forme d'une seule requête ou en faisant l'union de K-M requêtes.

Exemple 3.17 : Résultat de l'enrichissement de Q_1 sous forme d'une seule requête (Q_1^+)

```
SELECT V.idV
FROM VOYAGE V, DEPART D, TRANSPORT T
WHERE V.idD = D.idD AND D.date = '10/11/2007' AND
V.lieuArr = 'Madrid' AND V.nbJours = 3 AND
V.idT = T.idT AND V.lieuDep = 'Toulouse' AND T.moyen='avion' AND
( (D.heure > 22hOO AND T.direct=VRAI) OR
(D.heure > 22hOO AND V.prix<1000) OR
(T.direct=VRAI AND V.prix<1000) );
```

***Exemple 3.18* :** Résultat de l'enrichissement de Q_1 sous forme d'union de requêtes

```
SELECT V.idV
FROM ((SELECT V.idV
FROM VOYAGE V, DEPART D, TRANSPORT T
WHERE V.idD = D.idD AND D.date = '10/11/2007' AND
V.lieuArr = 'Madrid' AND V.nbJours = 3 AND
V.idT=T.idT AND V.lieuDep = 'Toulouse' AND T.moyen='avion'
AND D.heure > 22hOO AND T.direct=VRAI)
UNION ALL
    (SELECT V.idV
    FROM VOYAGE V, DEPART D, TRANSPORT T
    WHERE V.idD = D.idD AND D.date = '10/11/2007' AND
    V.lieuArr = 'Madrid' AND V.nbJours = 3 AND
    V.idT=T.idT AND V.lieuDep = 'Toulouse' AND T.moyen='avion'
    AND D.heure > 22hOO AND V.prix<1000)
        UNION ALL
        (SELECT V.idV
        FROM VOYAGE V, DEPART D, TRANSPORT T
        WHERE V.idD = D.idD AND D.date = '10/11/2007' AND
        V.lieuArr = 'Madrid' AND V.nbJours = 3 AND
        V.idT=T.idT AND V.lieuDep = 'Toulouse' AND T.moyen='avion'
        AND T.direct=VRAI AND V.prix<1000) )
        GROUP BY V.idV
        HAVING count(*) > 2 ;
```

4. 2. Personnalisation à l'aide des langages

Si l'on veut faire une nette distinction entre la notion de profil et la notion de requête, il faudrait que les langages de requêtes soient à même d'intégrer certaines informations et préférences de profils.

Depuis quelques années, les systèmes de bases de données proposent des langages de requêtes avancées dont certains concepts sont très appropriés à la prise en compte des informations de profil. Ainsi, une extension intéressante de SQL (PREFERENCE-SQL) permet d'exprimer bon nombre de préférences et est de ce fait très adaptée à la personnalisation de l'information; ces préférences peuvent être numériques (AROUND, BETWEEN, LOWEST, HIGHEST) ou non-numériques (POS, NEG, POS/NEG, POS/POS, EXPLICIT).

La nouveauté dans PreferenceSQL est la clause **« PREFERING … »** qui permet l'utilisation des fonctions de préférence. La clause **« CASCADING »** quant à elle, permet l'expression des préférences hiérarchiques. Ces clauses utilisent des opérateurs

sur les préférences. Ces opérateurs ont permis à l'utilisateur dans les langages tel que preferenceSQL :

1. D'exprimer les critères approximatifs

- ***La préférence « Autour de » (AROUND)***

Exprime l'intérêt de l'utilisateur vers les éléments dont la valeur de l'attribut A est la plus proche possible de la valeur donnée en paramètre (Z).

- ***La préférence « Entre » (BETWEEN)***

Permet à l'utilisateur d'exprimer ses préférences pour les éléments dont la valeur pour un attribut donné A est comprise dans un intervalle. Les valeurs extrêmes (low, up) sont les meilleurs.

2. D'exprimer les critères qualitatifs (Highest, Lowest)

- ***Les préférences « Plus Petit » (LOWEST) et « Plus Grand » (HIGHEST)***

Cherchent la plus petite ou la plus grande valeur d'un attribut.

P est nommée ***lowest*** si : P = LOWEST (A) si $x <_P y$ ssi $x>y$

P est nommée ***highest*** si : P = HIGHEST (A) si $x <_P y$ ssi $x<y$

Exemple 3.19 Si un utilisateur cherche une machine caractérisée par une plus grande capacité mémoire et une plus grande vitesse de processeur, alors il exprime sa requête dans la syntaxe du langage de préférence comme suit :

```
SELECT * FROM computers
PREFERRING HIGHEST(main_memory)
AND HIGHEST(cpu_speed);
```

Dans le format de ce langage, toutes les conditions de la clause *WHERE* (appelées *hard conditions*) sont d'abord considérées. Ensuite, les clauses *PREFERRING* sont évaluées suivant le langage de préférence considéré pour prendre en compte les préférences (appelées *soft conditions*). La structure fondamentale d'un bloc type d'une requête *Preference SQL* est de la forme :

```
SELECT <selection>  FROM <table_references>
WHERE <hard_conditions>
PREFERRING <soft_conditions>
GROUPING <attribute_list>
BUT ONLY <but_only_condition>
ORDER BY <attribute_list>
```

3. D'ordonner les valeurs préférées d'un résultat d'une requête

Certains langages permettent de spécifier une hiérarchie de préférences établissant un ordre partiel entre les critères de sélection. Dans PEFERENCE-SQL [Kießling, 2002], cet ordre partiel est défini à l'aide de deux clauses complémentaires: la clause *PREFERING* qui fixe les choix initiaux et la clause *CASCADING* qui fixe les choix secondaires. Par exemple si l'utilisateur de l'exemple précédent, préfère descendre dans les hôtels 3 étoiles lors de ses voyages et en privilégiant ceux du centre ville, sa requête s'exprime de la façon suivante :

```
SELECT * FROM DESTINATION D, HOTEL H
WHERE V.idH = H.idH AND prix ARROUND 600
PREFERING H.nombre_étoiles = 3
CASCADING H.région = 'centre ville'
```

La clause WHERE est exécutée sans l'influence des préférences. Ensuite la clause PREFERING est appliquée, suivie des clauses CASCADING qui sont exécutées les unes après les autres. Si, en appliquant une clause, le résultat devient nul, son effet est annulé et elle n'est pas prise en compte.

Dans notre exemple si plusieurs tuples satisfont les critères de la requête définis dans la clause WHERE, ceux qui contiennent des hôtels de trois étoiles sont retenus et finalement parmi eux, on choisit les tuples contenant des hôtels au centre ville.

Bien que la notion de profil n'existe pas dans les approches d'extension des langages de requêtes, ces langages ouvrent la voie à la personnalisation de l'accès aux données. Une partie du profil utilisateur (principalement le centre d'intérêt) peut être automatiquement intégrée aux requêtes par un processus de réécriture et ainsi simplifier le processus d'écriture de la requête par l'utilisateur.

L'enrichissement progressif des langages de requêtes permet à l'avenir d'en faire aussi des langages de définition de profils.

4. De rechercher les meilleurs résultats d'une requête

Les opérateurs : POS, NEG, POS/NEG, POS/POS, EXPLICIT [Kießling 2002] permettent l'ordonnancement des valeurs préférées.

Prenons par exemple une vue VOYAGE (Destination, Transport). La préférence des voyages en train par rapport à ceux en car peut être définie par l'expression : (d, train)>(d, car) pour une destination d. La même préférence peut être exprimée en utilisant les ensembles de valeurs préférées:

(Transport, POS1-set{train}, POS2-set{car}). Ici les meilleures valeurs pour l'attribut ″Transport″ sont celles contenues dans le premier ensemble positif (POS1-set). L'ensemble POS2- set contient les valeurs du même attribut qui sont moins préférées que celles du premier ensemble positif, mais plus pertinentes que toutes les autres valeurs qui ne figurent dans aucun des deux ensembles.

4. 2. 1. Les extensions des langages pour supporter la personnalisation

L'idée principale de la personnalisation est de faciliter l'écriture des requêtes de l'utilisateur en les enrichissant avec des données invariantes communes à toutes les requêtes pour un utilisateur donné. Dans ce contexte, l'utilisateur doit être capable d'exprimer toutes ses préférences et exigences de façon simple. Ces exigences doivent être traduites dans un langage afin d'être ajoutées aux requêtes et exécutées. Les langages classiques comme SQL ont une syntaxe précise et ne supportent pas la notion de préférence. Pour cette raison, ces dernières années ont vu naître des extensions du langage qui supportent l'évaluation de fonctions externes ou utilisent d'autres clauses permettant d'exprimer des préférences. Parmi ces approches on retrouve des opérateurs complexes comme l'opérateur *Winnow* (Best) [Chomicki 2002] ou l'opérateur Skyline [Borzsonyi et al, 2001] et l'ajout de la nouvelle clause prefer [Lacroix 1987].

- ***L'opérateur Skyline***

L'opérateur Skyline [Borzsonyi et al, 2001] est définit dans le contexte d'une base de données relationnelle interrogeable par le langage SQL. Les éléments du *skyline* sont ceux qui ne sont pas dominés par d'autres éléments. Un élément domine un autre s'il est au moins aussi bon que le dominé dans toutes ses dimensions et meilleur dans au moins une dimension. Un élément peut être vu comme un point dans un espace à m dimensions où m est le nombre d'attributs qu'il possède. Une des propriétés intéressantes du skyline est que pour toute fonction monotone f : M→R si p maximise cette fonction, alors p est contenu dans le skyline (le poids des critères n'est pas important car tout maximum est contenu dans le skyline) et en plus pour chaque point du skyline il existe une fonction monotone qu'il maximise (il ne contient pas de points non-pertinents). Pour son intégration on étend le langage SQL avec une clause optionnelle dont la syntaxe est la suivante :

SKYLINE OF [DISTINCT] d_1 [MIN | MAX | DIFF], ..., d_m [MIN | MAX | DIFF] où les d_i sont les dimensions du skyline (attributs de la relation).

- ***L'opérateur Winnow (Best)***

L'opérateur Winnow [Chomicki, 2002] (appelé Best Match Only dans [Kießling, 2002]) permet la sélection des éléments préférés d'une relation (R) par rapport à une formule de préférence (C), il est défini dans le contexte des bases de données relationnelles. De cette façon, une requête avec des préférences est une requête de l'algèbre relationnelle contenant au moins une occurrence de l'opérateur Winnow. Si une relation de préférence sur un schéma relationnel est définie en utilisant une formule de préférence qui est un ordre partiel strict, alors pour toute instance non vide et finie de ce schéma l'opérateur Winnow renvoie un résultat non vide.

- ***La clause prefer :***

Dans l'approche de [Chomicki, 2002], les préférences de l'utilisateur sont décrites par une clause "*prefer*" qui s'ajoute au langage de requête habituel. Il n'y a pas vraiment de profil stable de l'utilisateur. Les clauses de préférence peuvent être utilisées plusieurs fois dans la même requête. Ils existent deux cas possibles: préférences ayant la même importance et préférences imbriquées. Lorsque l'on évalue des préférences d'importances différentes ou préférences imbriquées, on se sert de la syntaxe "*from witch prefer...*" en considérant que cette clause est moins importante que la précédente. Le deuxième cas possible est la définition de préférences de même importance. Leur expression est faite par la répétition de la clause prefer. Le résultat est formé par les tuples qui satisfont un nombre maximal de préférences.

Le tableau ci-dessous résume la personnalisation et le profil considérés dans les travaux étudiés :

Travaux de	***Approche***	***Préférence sur***	***Technique***
Kießling	Quantitative/ Qualitative	Tuples /Attributs	Implémentation d'une extension du langage SQL
Chomicki	Qualitative	Tuples	/
Koutrika and Ioannidis	Quantitative	Requêtes	Enrichissement de requêtes

Tab. 3.2 – Comparaison des différents travaux de personnalisation

5. Contrainte d'optimisation pour les requêtes personnalisées

La contrainte liée à la personnalisation de la requête (Constraint of Query Personalization : CQP) est une approche intégrée à l'exécution d'une requête de la base de données qui tient compte dynamiquement des requêtes, la requête optimale construite doit satisfaire les contraintes liées aux paramètres qui la caractérisent à savoir :

Le degré d'intérêt (doi), qui doit être maximal ou posséder une borne inférieure, du fait que le but de la personnalisation d'une requête est de produire une réponse intéressante à l'utilisateur.

Le coût d'exécution (cost), qui doit être minimal ou posséder une borne supérieure, ce qui le définit par nature.

La taille de la requête (size), qui doit être comprise entre une borne supérieure et une borne inférieure, du fait que le but de la personnalisation et de réduire la taille de la réponse (borne supérieure) et d'éviter les réponses vides (borne inférieure), Le tableau suivant récapitule toutes les contraintes possibles liées à ces paramètres et par conséquent chaque ligne du tableau représente un problème d'optimisation lié à la requête :

Problème	**doi**	**Cost**	**Size**
1	MAX		Smin ≤ Size ≤ Smax
2	MAX	Cost ≤ Cmax	
3	MAX	Cost ≤ Cmax	Smin ≤ Size ≤ Smax
4		MIN	Smin ≤ Size ≤ Smax
5	Doi ≥ dmin	MIN	
6	Doi ≥ dmin	MIN	Smin ≤ Size ≤ Smax

Tab. 3.3 – Problèmes des contraintes d'optimisation des requêtes personnalisées (CQP) [Koutrika et Ioannidis 2005b]

Le problème d'optimisation de requêtes est un problème de maximisation d'intérêt ou de minimisation de coût.

En outre, le choix du problème CQP à résoudre dépend du contexte de la requête [Koutrika et Ioannidis 2005b].

Dans le paragraphe suivant, nous donnons une récapitulation de : comment ces contraintes sont prises en compte dans les différents travaux de personnalisation déjà présentés :

- La personnalisation de requête proposée dans Koutrika et Ioannidis est un problème d'optimisation dont le but est de maximiser l'intérêt de l'utilisateur dans la réponse à la requête en se basant sur le profil. On note que dans cette méthode la réponse peut être vide puisque la personnalisation est faite sans accès aux données.

En plus des préférences exprimées directement par l'utilisateur, [Koutrika and Ioannidis, 2005a] propose de prendre en compte des contraintes telles que le coût d'exécution et/ou la taille de la réponse. L'objectif est de construire une requête personnalisée Q_U qui est optimale par rapport à un paramètre de requête et qui satisfait des contraintes sur d'autres paramètres. Il s'agit d'un problème d'optimisation avec contraintes visant à maximiser l'intérêt de l'utilisateur pour les résultats d'une requête posée sous contraintes de coût d'évaluation et/ou de taille du résultat :

$doi(Q_U) = MAX\{ doi(Q_X) | Q_x = Q \wedge P_x, P_x \subseteq P, Q_x$ satisfait les contraintes$\}$

et qui minimise le coût d'exécution de la requête sous contraintes sur l'intérêt et/ou la taille :

$cost(Q_U) = MIN\{ cost(Q_X) | Q_x = Q \wedge P_x, P_x \subseteq P, Q_x$ satisfait les contraintes$\}$.

- Dans la méthode de personnalisation proposée par Chomicki, aucune contrainte n'est prise en compte.

6. Conclusion

A travers ce chapitre nous avons constaté que la personnalisation dans le domaine des bases de données a été envisagée sous deux aspects :

(i) L'utilisation du profil de l'utilisateur pour l'enrichissement de la requête. Et

(ii) L'utilisation des langages d'expression des requêtes, l'objectif des chercheurs dans ce domaine est la définition d'un langage universel, ressemblant à SQL. Cette approche a permis à l'utilisateur d'exprimer toutes ses préférences, mais n'utilise pas la notion de profil. L'utilisateur est contraint d'écrire à chaque fois la requête complète qui définit son besoin d'information ce qui est un inconvénient non négligeable.

L'objectif de la personnalisation est d'éviter à l'utilisateur d'écrire à chaque fois la partie commune à ses requêtes, l'élaboration d'un modèle de profil générique est le premier pas vers la construction de systèmes natifs de personnalisation capables de rechercher et d'extraire des informations de qualité selon les préférences d'un utilisateur. C'est pour cette raison que les chercheurs intègrent le profil de l'utilisateur en le modélisant sous forme d'un modèle de préférence.

Ce modèle représente le noyau de la personnalisation qui consiste à modéliser le profil de l'utilisateur sous forme d'un graphe puis à partir de ce modèle, on peut faire ressortir, initialement, les top K préférences du profil et les intégrées dans la requête initiale. Ou bien exploiter ces préférences dans des extensions des langages de requêtes.

La première partie de l'état de l'art étant terminée, cette partie consiste en une initiation à la personnalisation. Nous entamerons dans le chapitre suivant la personnalisation des requêtes décisionnelles.

4 *La personnalisation dans les entrepôts de données*

Si la personnalisation n'est pas une idée nouvelle dans les domaines précédemment évoqués, elle constitue un axe de recherche émergent dans le domaine des entrepôts de données. L'intérêt de cet axe de recherche peut être motivé à la fois vis-à-vis de la volumétrie des données connue pour être importante dans les entrepôts de données et du rôle central que joue l'utilisateur dans le processus décisionnel. En effet, il est en interaction directe avec le système au niveau de l'analyse des données, en particulier dans le contexte de la navigation. Différentes pistes ont d'ores et déjà été initiées.

Avant de détailler les travaux réalisés sur la personnalisation dans les entrepôts de données, nous consacrons la première partie de ce chapitre à situer le contexte de nos travaux de recherche, à savoir les entrepôts de données, en donnant d'abord les concepts généraux qui caractérisent les entrepôts de données, leur objectif, leur modélisation, leur exploitation, leur mise en œuvre, etc.

Une fois ces concepts généraux présentés, nous dressons un panorama des travaux sur la personnalisation dans les entrepôts de données. Puis, nous présentons en détail le problème de la personnalisation des requêtes OLAP utilisées pour interroger un ED.

1. Contexte de notre travail : Entrepôts de données, conception et exploitation

Avant de s'intéresser justement au problème de la personnalisation dans les entrepôts de données, nous nous devons d'évoquer certaines généralités sur ces derniers afin d'éclairer nos propos futurs. Il s'agit en effet de revenir sur ce que sont les entrepôts de données, comment ils sont conçus, quelles sont leurs caractéristiques, etc.

1. 1. Vocation des entrepôts de données

Un entrepôt de données est conçu pour répondre à un ensemble de besoins d'analyse communs à la plupart des utilisateurs. Cependant, les utilisateurs peuvent avoir des besoins variés auxquels l'entrepôt n'est pas forcément en mesure de répondre, à fortiori dans une grande entreprise, dans laquelle les utilisateurs exercent de nombreux métiers. La création de *magasins de données* tente de se rapprocher des besoins utilisateurs en fonction de leurs métiers. Néanmoins, chaque utilisateur dispose de connaissances particulières du domaine et des besoins d'analyse qui lui sont propres.

L'entrepôt de données permet avant tout d'intégrer des sources de données. Mais son objectif final n'est autre que de supporter un processus d'analyse en ligne. Par rapport aux aspects d'intégration et d'analyse, l'entrepôt se trouve finalement être au cœur de l'architecture décisionnelle dont l'objectif est de construire de l'information utile à l'aide à la décision.

1. 2. Architecture décisionnelle

L'entrepôt de données est le support nécessaire à la réalisation d'une architecture qualifiée de décisionnelle, qui va permettre, comme l'indique son nom, l'aide à la décision grâce au processus d'analyse qu'elle offre.

Cette architecture décisionnelle, peut être représentée classiquement selon le schéma de la figure 4.1. On peut y distinguer la partie *sources*, la partie *gestion des données* et enfin la partie *analyse*. On parle généralement d'architecture n-tiers en raison des différentes couches possibles pour gérer les données (entrepôt, magasin, etc.) et réaliser des analyses.

Cette architecture met en avant deux phases caractéristiques que sont l'intégration des données et l'analyse. Ces phases sont basées sur cinq éléments essentiels qui composent le système décisionnel [Favre, 2007] :

– ***Sources de données***: Ce sont les bases de production (relevant d'un système OLTP), les fichiers qui correspondent à des sources internes; mais elles peuvent également être d'origine externe (Internet, bases de partenaires, etc.);

– ***Entrepôt de données***: C'est le lieu de stockage massif et centralisé des informations utiles pour les décideurs; il est associé à un ensemble de méta-données (informations sur les données) qui forme en quelque sorte un référentiel de données contenant différentes informations telles que les règles de transformation et de nettoyage des données permettant d'assurer différentes tâches telles que la maintenance de l'entrepôt.

– ***Magasins de données***: Ce sont des extraits de l'entrepôt orientés métiers, activités (on parle de données verticalistes), contenant un volume moindre de données, permettant alors des analyses plus rapides;

– ***Cubes de données*** : Ce sont des contextes d'analyse multidimensionnels;

– ***Outils d'analyse*** : Ils permettent de manipuler les données et de les analyser.

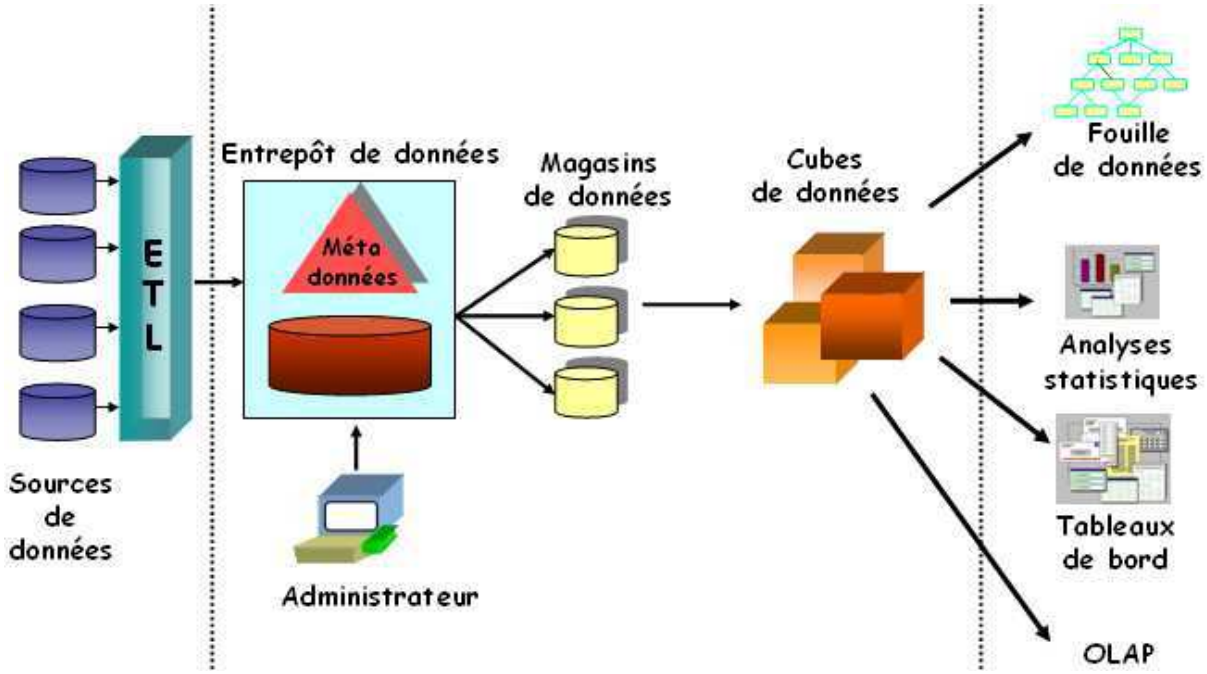

Fig. 4.1 – Architecture décisionnelle

1. 2. 1. Intégration de sources de données

Un entrepôt de données constitue avant tout une alternative pour l'intégration de diverses sources de données. Un système d'intégration a pour objectif d'assurer à un utilisateur un accès à des sources multiples, reparties et hétérogènes, à travers une interface unique. En effet, l'avantage d'un tel système est que l'utilisateur se préoccupe davantage de ce qu'il veut obtenir plutôt que comment l'obtenir, l'objectif étant l'obtention d'informations. Ainsi, cela le dispense de tâches telles que chercher et trouver les sources de données adéquates, interroger chacune des sources de données en utilisant sa propre interface et combiner les différents résultats obtenus pour finalement disposer des informations recherchées.

On trouve ensuite la phase d'analyse qui peut exploiter aussi bien des analyses statistiques, des tableaux de bord, de la fouille de données, de l'analyse en ligne, le couplage des deux derniers [Missaoui et al, 2007], etc.

Notons que ces processus d'analyse peuvent s'opérer aussi bien sur l'entrepôt de données, que sur les magasins de données, ou encore sur les cubes de données (ce concept sera définit dans ce qui suit).

Nous revenons plus en détail sur ces deux phases en abordant tout d'abord la modélisation de l'entrepôt de données qui constitue le résultat de l'intégration, puis en abordant le processus d'analyse de façon plus détaillée.

1. 2. 2. Définition d'un entrepôt de données

En 1996, Bill Inmon [Inmon, 96] définit un entrepôt de données comme étant une « Collection de données orientées sujet, intégrées, non volatiles et historisées, organisées pour le support du processus d'aide à la décision ».

Les données sont « *orientées sujet* » dans la mesure où elles sont organisées par thèmes, l'entrepôt de données est organisé autour des sujets majeurs et des métiers de l'entreprise. Il permet une vision transversale des différentes activités de l'entreprise.

Le fait que les données soient *« intégrées »* exprime leur provenance de sources différentes. Cette intégration nécessite une bonne connaissance des sources de données, des règles de gestion, de la sémantique des données, etc.

En outre, les données sont *« historisées »* afin de rendre possible la réalisation d'analyses au cours du temps, nécessitant un recours à un référentiel temporel associé aux données.

De plus, les données sont dites *« non volatiles »*. Cela signifie que les données stockées au sein de l'entrepôt de données ne peuvent pas être supprimées. Une requête émise sur les mêmes données à différents intervalles de temps doit donner le même résultat. Cela doit permettre de conserver la traçabilité des informations et des décisions prises.

Enfin, les données sont *« organisées pour le support du processus d'aide à la décision »*; il s'agit en l'occurrence d'une organisation multidimensionnelle. Cette organisation est en effet propice à l'analyse et, en particulier, à l'agrégation, comme nous le montrerons par la suite.

1. 3. Concepts de base et modélisation des entrepôts de données

1. 3. 1. La modélisation multidimensionnelle

Les modèles de conception des systèmes transactionnels OLTP ne sont pas adaptés aux systèmes OLAP dont les requêtes sont souvent très complexes, utilisent beaucoup de jointures, demandent beaucoup de temps de calcul et sont de nature ad hoc. Pour ce type

d'environnement OLAP, une nouvelle approche de modélisation a été proposée : *la modélisation multidimensionnelle*.

La modélisation multidimensionnelle est une technique de conceptualisation et de visualisation des modèles de données. Elle permet une structuration des données et offre des vues de données facilitant leur analyse.

Une modélisation multidimensionnelle a pour principal objectif d'avoir une vision multidimensionnelle des données. Les données sont organisées de manière à mettre en évidence le sujet analysé et les différentes perspectives de l'analyse. Une modélisation multidimensionnelle est intéressante d'une part parce qu'elle représente une modélisation assez proche de la manière de penser des analystes, et d'autre part, elle facilite aux utilisateurs la compréhension des données. Le modèle multidimensionnel renferme les concepts de *fait* et de *dimension*.

- Un ***fait*** représente le sujet ou le thème analysé. Il présente un centre d'intérêt de l'entreprise et est considéré comme un concept clé sur lequel repose le processus de prise de décision. Un fait est formé de « *mesures* » ou attributs du fait (atomiques ou dérivés) qui correspondent aux informations liées au thème analysé. Les mesures sont stockées dans des tables de faits qui contiennent les valeurs des mesures et les clés vers les tables de *dimensions*.

- Une ***dimension*** représente un contexte d'analyse d'un fait et se présente sous forme d'une liste d'éléments organisés de façon hiérarchique. Par exemple, pour la dimension *temps*, nous pouvons avoir la hiérarchie suivante : *année, semestre, trimestre, mois, semaine et jour*. Les dimensions sont caractérisées par des attributs de dimensions. Elles sont stockées dans des tables de dimensions qui contiennent *les niveaux hiérarchiques* des dimensions ainsi que les formules à appliquer sur les données numériques pour passer d'un niveau à un autre. Une hiérarchie est dite complète si tous les objets d'un niveau de la hiérarchie appartiennent à une seule classe d'objets d'un niveau supérieur et forment cette classe.

Le principal constructeur des modèles multidimensionnels est le *cube* de données. Un cube de données est composé de *cellules* qui représentent les mesures (les attributs du fait). Le cube ci-dessous permet d'analyser les mesures selon les différentes dimensions : *produit, site et temps*. Les hiérarchies définies sur une dimension peuvent être simples ou multiples.

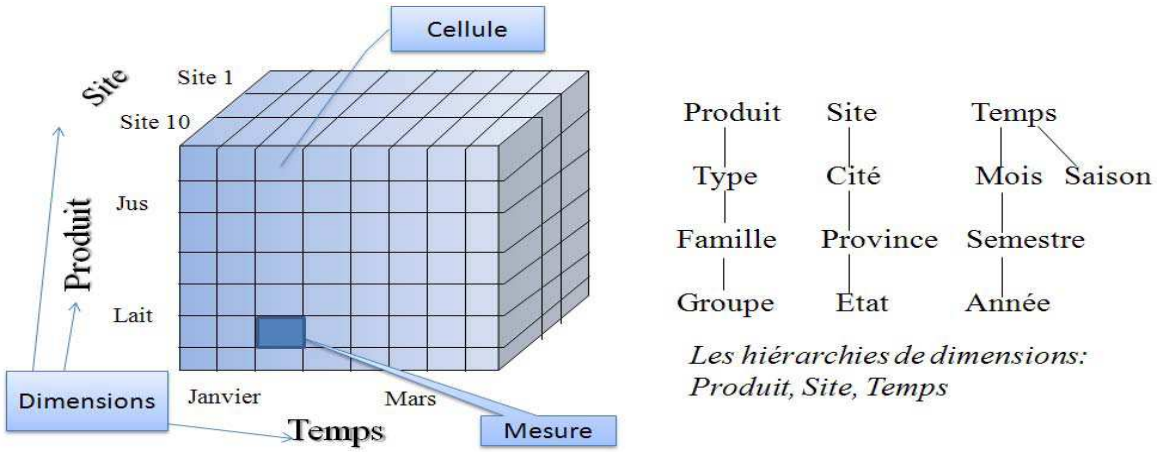

Fig. 4.2. – Le cube multidimensionnel Vente

1. 3. 2. Importance des hiérarchies de dimension

Si un des objectifs de l'analyse en ligne est bien entendu la rapidité des temps de réponse, la richesse des possibilités d'analyse a également son importance. Cette richesse dépend du schéma de l'entrepôt et, plus particulièrement, des dimensions et de leur(s) hiérarchie(s). En effet, la navigation dans les données est conditionnée par cette organisation dimensionnelle des données. Cette navigation se base entre autres sur l'agrégation des données. Celle-ci est soutenue par le concept de hiérarchie. En effet, dans les entrepôts de données, les hiérarchies vont permettre de représenter comment doivent être agrégées les données. La hiérarchisation des données dans les modèles multidimensionnels permet des analyses à différents niveaux de détail. Classiquement, les hiérarchies sont représentées par des concepts qui sont reliés par des relations un à plusieurs. Autrement dit, une instance d'un niveau inferieur correspond à une seule instance du niveau supérieur et une instance du niveau supérieur correspond à plusieurs instances du niveau inferieur. [Favre, 2007]

1. 3. 3. Les implémentations des modèles multidimensionnels

Ces concepts de base ont permis de définir trois schémas classiques reconnus comme relevant d'un niveau logique de conception, en raison du recours à la notion de table (table de faits, table de dimension).

1. ***Le schéma en étoile*** : une table de faits centrale en BCNF[1] contenant les faits à analyser, est reliée aux tables de dimensions par des clefs étrangères. Chaque dimension est décrite par une seule table [Kimball, 1996]. Le schéma en étoile est à la base des deux modèles suivants.
2. ***Le schéma en flocon*** : il représente la table des faits selon le même principe que dans le schéma en étoile, mais les tables de dimensions sont normalisées afin de

[1] BCNF : c'est la Forme Normale de *Boyce-Codd*.

représenter plus explicitement les hiérarchies [Jagadish et al., 1999] et pour réduire les redondances.

Remarque

La normalisation permet un gain d'espace de stockage en évitant la redondance de données, mais engendre une dégradation des performances, dans la mesure où elle multiplie le nombre de jointures à effectuer pour l'analyse. Le troisième et dernier schéma est le schéma en constellation, aussi appelé flocon de faits. Ce schéma fait coexister plusieurs tables de faits qui partagent ou pas des dimensions communes (hiérarchisées ou non).

3. ***Le schéma en constellation de faits*** : plusieurs tables de faits partagent les tables de dimension [Chaudhuri and Dayal, 1997]. Les données ainsi organisées sont sous forme de cubes multidimensionnels. Un cube est une représentation calculée à partir des données de l'entrepôt. Le cube présente les mesures de l'analyse, élémentaires ou agrégées, en fonction des différents niveaux de granularité des dimensions.

Exemple 4.1 Nous pouvons modéliser les données de ventes en utilisant un cube qui est décrit sous forme de schéma en étoile par la Fig 4.3.

Ce cube, nommé ″SalesCube″, est composé d'une table de faits ″Sales″ et de quatre dimensions qui sont ″Time″, ″Product″, ″Location″ et ″Measures″. Chaque cellule de ce cube stocke la valeur d'une mesure donnée pour un produit particulier sur une localisation particulière pour une période donnée. Nous pouvons, par exemple, avoir la hiérarchie suivante pour la dimension localisation : all → country → région → city

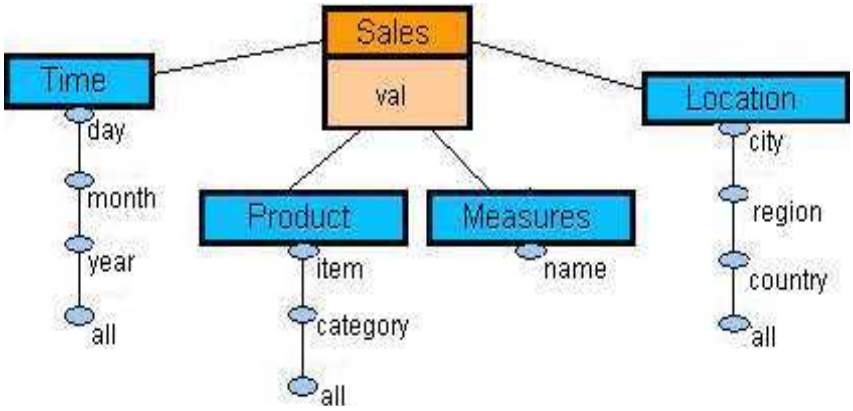

Fig. 4.3 – Le cube SalesCube en schéma en étoile [Golfarelli and Rizzi, 1998].

Dans le contexte OLAP, la communauté des bases de données s'intéresse à la définition d'une modélisation logique en prenant en compte la nature multidimensionnelle et hiérarchisée des données. Plusieurs modèles multidimensionnels formels et langages d'interrogation correspondants ont été proposés [Romero and Abelló, 2007], où l'objectif majeur est la vision multidimensionnelle des données pour faciliter les prises de décisions.

Cependant, chaque approche présente une vision particulière des besoins, de la terminologie et du formalisme de l'analyse multidimensionnelle. En conséquence, il n'y a aucun modèle multidimensionnel formel consensuel établi.

1. 4. Exploitation de l'entrepôt

L'entrepôt a pour objectif final l'analyse des données en vue de la prise de décision. Différents types d'analyse peuvent être réalisés : analyses statistiques, fouille de données, etc. Mais on peut également parler d'analyse en ligne OLAP. Il s'agit en l'occurrence d'une navigation dans les données. Cette analyse peut être qualifiée d'exploratoire. Le principe général est d'arriver au cours de la navigation à détecter des points intéressants qu'on essaye de décrire, d'expliquer en naviguant, par exemple en allant chercher davantage de détails. Le rôle de l'utilisateur est ici central puisque c'est lui qui réalise la navigation, celle-ci nécessite une connaissance du domaine afin d'être en mesure de savoir si les valeurs des mesures sont intéressantes ou non.

Afin de réaliser la navigation, différents operateurs s'appliquent au niveau d'un cube de données. Ils peuvent être classés en deux catégories : les operateurs liés à la structure et les opérateurs liés à la granularité.

1. 4. 1. Opérateurs liés à la structure

D'une manière générale, les operateurs liés à la structure permettent la manipulation et la visualisation du cube.

Le tableau ci-après résume la description de chaque opérateur.

Opération	***Traduction***	***Description***
Rotate	Pivot	Consiste à faire effectuer à un cube une rotation autour d'un de ses trois axes passant par le centre de deux faces opposées, de manière à présenter un ensemble de faces différent. C'est en quelque sorte une sélection de faces et non des membres.
Switch	Permutation	Consiste à inter-changer la position des membres d'une dimension.
Split	Division	Consiste à présenter chaque tranche du cube et à passer d'une présentation tridimensionnelle d'un cube à sa présentation sous la forme d'un ensemble de tables. Sa généralisation permet de découper un hypercube de dimension 4 en cubes.
Nest	Emboîtement	Permet d'imbriquer des membres à partir du cube. L'intérêt de cette opération est qu'elle permet de grouper sur une même représentation bi-dimensionnelle toutes les informations (mesures et membres) d'un cube, quel que soit le nombre de ses dimensions.
Push	Enfoncement	Consiste à combiner les membres d'une dimension aux mesures du cube, i.e. de faire passer des membres comme contenu de cellules.

Tab. 4.1 – Opérations OLAP liées à la structure.

1. 4. 2. Opérateurs liés à la granularité

Quant aux opérations liées à la granularité, il s'agit d'agréger les données pour obtenir des données résumées et inversement. Les opérateurs liés à la granularité sont au nombre de deux (Tab 4.2). Ils se basent sur la hiérarchie de navigation entre les différents niveaux, et sont présentés dans le tableau suivant.

Opération	***Traduction***	***Description***
Roll-up	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. Une fonction d'agrégation (somme, moyenne, etc.) en paramètre de l'opération indique comment sont calculées les valeurs du niveau supérieur à partir de celles du niveau inférieur.
Drill-down	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.

Tab. 4.2 – Opérations OLAP liées à la granularité

1. 5. Mise en œuvre

Afin de mettre en œuvre l'entreposage des données et l'exploitation de ces dernières, différentes alternatives sont envisageables. Ces alternatives se nomment ROLAP (Relational OLAP), MOLAP (Multidimentional OLAP), HOLAP (Hybrid OLAP), DOLAP (Desktop OLAP). Elles sont liées à l'environnement physique d'implantation.

Afin de classifier une solution OLAP parmi les alternatives évoquées, il existe deux critères: la technologie de stockage des données d'une part et les techniques de traitement de ces données d'autre part. Concernant la technologie de stockage, les trois possibilités sont les suivantes :

- *Base de données relationnelle:* les données sont stockées dans un SGBD relationnel. Il permet un stockage presque infini des données (ROLAP).
- *Base de données multidimensionnelle (Cube)*: les données sont stockées dans une base de données multidimensionnelle le plus souvent propriétaire. Dans ce cas, il y'a des limitations quant à la quantité des données traitées (MOLAP).
- *Fichiers sur le poste client:* une petite quantité de données (mini-base multidimensionnelle ou extraction de cube) est stockée directement sur le poste client de l'utilisateur (DOLAP).

En ce qui concerne les techniques de traitements des données, on distingue trois solutions :

– *SQL*: SQL est utilisé pour effectuer les différents traitements sur les données.

On réalise les opérations de traitement (forage vers le haut, vers le bas, etc.) en utilisant des requêtes en général très complexes et très exigeantes en terme de ressources et de temps d'exécution. Il s'agit de l'alternative relationnelle (ROLAP).

– *Serveur de traitement OLAP*: il s'agit de l'approche la plus adaptée aux traitements de données. Un serveur, conjointement avec la base de données, est alors dédié à effectuer les différents traitements de données. Dans ce cas, les performances sont généralement très bonnes. Il s'agit de l'alternative multidimensionnelle « pure » (MOLAP).

– *Client de traitement OLAP*: un nombre limite de traitements OLAP se font sur le poste client de l'utilisateur. Il s'agit de l'alternative « bureau », en local (DOLAP).

En se basant sur ces deux critères, on peut alors différencier aisément les alternatives en combinant une technologie de stockage et une technique de traitement.

Les combinaisons sont regroupées dans le Tableau 4.3.

Stratégie	***Stockage des données***	***Traitement des données***
MOLAP	Base de données dimensionnelle	Serveur de traitement OLAP
ROLAP	Base de données relationnelle	SQL avancé
DOLAP	Fichier sur le poste client	Client de traitement OLAP
HOLAP	MOLAP pour données sommaires & ROLAP pour données détaillées	

Tab. 4.3 – Alternatives de mise en œuvre OLAP

2. La personnalisation dans les entrepôts de données

La personnalisation dans le cadre des entrepôts de données présente un réel intérêt dans un contexte où les analyses pour permettre la prise de décision sont réalisées par l'utilisateur lui-même. Ce dernier a sa propre connaissance des données visualisées lors de la navigation et donc des besoins qui peuvent lui être propre, d'où cet intérêt pour la personnalisation.

Autrement dit, l'utilisateur est en interaction directe avec le système au niveau de l'analyse des données, en particulier dans le contexte de la navigation. Différentes pistes ont d'ores et déjà été initiées.

Le paradigme d'accès à l'information est basé sur la formulation de requêtes (Fig 4.4).

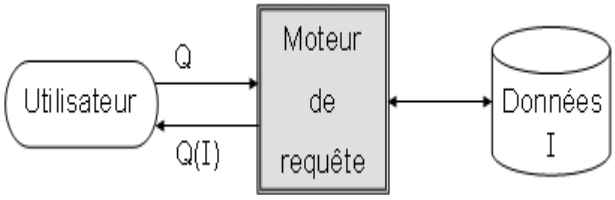

Fig. 4.4 – Architecture d'un système d'interrogation de données.

Avant de détailler les travaux recensés, nous donnons d'abord la différence entre personnalisation et recommandation qui se prêtent souvent à confusion.

2. 1. Personnalisation et recommandation

La recommandation fut parmi les premières approches suivies pour fournir un accès personnalisé à l'utilisateur. Un système de recommandation a pour but de suggérer à l'utilisateur un contenu informationnel susceptible de répondre à ses besoins.

La recommandation est un service de personnalisation qui consiste à proposer à l'utilisateur des éléments vis à vis de ses préférences ou en se servant de l'expérience des autres utilisateurs.

Les deux processus de recommandation et de personnalisation de l'information diffèrent sur le contenu de la réponse :

- La personnalisation permet de réduire le contenu du résultat qui consiste en un sous ensemble de la réponse à la requête initiale posée par l'utilisateur,
- La recommandation permet de compléter le résultat personnalisé. Elle consiste à enrichir le contenu du résultat avec de nouvelles informations qui ne sont pas demandées par la requête initiale de l'utilisateur. Ainsi, elle garantit une réponse non vide qui est possible dans la personnalisation.

Quel que soit le domaine technologique, la personnalisation de l'information peut être exploitée selon deux modes de gestion : en recommandation ou en interrogation.

- ✓ Les systèmes de recommandation [Giacometti et al, 2008] exploitent les profils des utilisateurs ou de communautés d'utilisateurs pour disséminer des offres ciblées sur les centres d'intérêt et les préférences de ces derniers.
- ✓ La personnalisation en interrogation [Bellatreche et al, 2006] consiste à adapter l'évaluation d'une requête par rapport aux caractéristiques et aux préférences de l'utilisateur qui l'a émise. Dans ce contexte, le système réagit à une demande spécifique de l'utilisateur en transformant sa requête afin de la rendre plus précise, en personnalisant l'affichage des résultats

2. 2. Travaux réalisés

La personnalisation dans les entrepôts de données peut être vue sous trois angles :

(i) Personnalisation de schéma ou bien de l'évolution du schéma de l'entrepôt [Favre, 2007] ou

(ii) Personnalisation de la navigation [Ravat et Teste 2008].

(iii) Personnalisation de la visualisation du résultat d'une requête OLAP [Bellatreche et al, 2006].

L'objectif de notre travail s'inscrit dans le troisième type de personnalisation, en effet, les travaux effectués sur la recommandation ainsi que sur la personnalisation de la navigation seront présentés brièvement. Cependant une présentation détaillée sera réservée au dernier type de personnalisation.

2. 2. 1. Travaux de [Favre 2007].

Afin d'enrichir les possibilités d'analyse d'un entrepôt de données, une proposition permettant la création de niveaux d'analyse supplémentaires dans les hiérarchies de dimension ou définissant de nouvelles hiérarchies de dimension a été faite dans [Favre, 2007]. Cette approche permet aux utilisateurs d'intégrer leurs propres connaissances sur la façon d'agréger les données sous la forme de règles de type « si-alors ». Une évolution du modèle de l'entrepôt permet le partage des chemins d'agrégation nouvellement créés entre les utilisateurs du système. Supposons qu'un utilisateur veuille analyser les ventes du produit « *food* », non pas par « *year* » ni par « *day* » et non plus par « *month* », mais par semaine, information qui n'est pas présente dans l'entrepôt. L'utilisateur peut alors exprimer ses connaissances sur les types de Time, afin de créer un niveau « *semaine* », correspondant à une nouvelle hiérarchie de la dimension « *Time* ». L'utilisateur pourra ainsi réaliser des analyses des ventes du produit « *food* » par semaine.

2. 2. 2. Travaux de [Giacometti et al, 2008]

Une session d'analyse OLAP peut être définie comme une session interactive durant laquelle un utilisateur lance une requête pour naviguer au sein du cube. Très souvent, le choix de la partie du cube à explorer et donc la requête correspondante à cette partie est une tâche difficile.

[Giacometti et al, 2008] Proposent un système de recommandation d'analyses multidimensionnelles en se basant sur la navigation qu'effectue un utilisateur donné par rapport aux navigations réalisées par les autres utilisateurs.

La proposition de Giacometti et al. (2008) permet la recommandation de requêtes pour anticiper sur une séquence de requêtes d'un utilisateur grâce à l'analyse des historiques de navigations réalisées par les autres utilisateurs. Par exemple, supposons qu'un utilisateur a réalisé une analyse des ventes par année et par country, puis avec un forage vers le bas par année et par région, et enfin par année (avec un second forage vers le bas). Si un nouvel utilisateur réalise une analyse des ventes par année et par country, puis une analyse par année

et par région, une analyse par année et par city lui sera recommandée, sa navigation étant similaire à une navigation réalisée précédemment.

2. 2. 3. Travaux de [Ravat et Teste 2008]

Ravat et Teste (2008) proposent une solution pour la personnalisation de la navigation OLAP en exploitant des préférences exprimées par des poids sur les éléments du schéma de la base de données multidimensionnelles, des poids reflétant l'intérêt de l'utilisateur.

Dans ce cas, l'utilisateur assigne des poids aux concepts multidimensionnels afin d'obtenir directement les analyses désirées, évitant ainsi des opérations de navigation.
Un système basé sur des règles ECA (Event, Condition, Action) permet de générer des tables multidimensionnelles contenant uniquement les données identifiées comme pertinentes en fonction des poids. Cette solution quantitative permet de simplifier l'expression des requêtes d'analyse.

Pour aller au-delà de cette approche quantitative, une solution de personnalisation qualitative est introduite par [Jerbi et al,.2008]

2. 2. 4. Travaux de [Jerbi et al,.2008]

Il s'agit non plus d'exploiter des poids, mais plutôt des ordres (représentation qualitative des préférences), ce qui rend la tâche plus aisée pour l'utilisateur. En outre, ces ordres 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 un autre. [Jerbi et al,.2009] poursuivent ces travaux en présentant un environnement OLAP intégrant des mécanismes de recommandation contextuelle des requêtes.

Nous avons préféré dresser une comparaison des différents travaux étudiés avant de détailler le travail de Bellatreche et al, qui est la base de notre inspiration. A noter que c'est l'unique travail qui traite la personnalisation de requêtes OLAP.

Références	*Définition du profil*		*Exploitation du profil*	
	Apprentissage	*Paramétrage*	*Transformation*	*Recommandation*
[Bellatreche et al., 2005]		Contraintes de Visualisation	Modification table résultat	
[Favre et al., 2007]		Règles si-alors	Modification schéma entrepôt	
[Giacometti et al., 2008]	Suite de requêtes			Requêtes suivantes
[Jerbi et al., 2008-2009]		Ordre des préférences selon contexte	Modification table résultat	Requêtes suivantes, enrichies, alternatives
[Ravat et al., 2008]		Poids sur les éléments du schéma	Modification table résultat	

Tab. 4.4. – Comparaison des travaux de personnalisation définis précédemment.

2. 2. 5. Travaux de [Bellatreche et al, 2005].

Ces travaux se concentrent sur la personnalisation des réponses aux requêtes utilisateur interrogeant les données du cube, c.-à-d. des visualisations. Dans cette approche, la visualisation est sous forme d'une table croisée.

L'objectif de ces travaux est de pouvoir fournir à l'utilisateur un résultat focalisé sur son centre d'intérêt, tout en prenant en compte des contraintes de visualisation.

Avant de détailler ces travaux, nous donnons d'abord les définitions formelles des concepts manipulés.

2. 2. 5. 1. Définition des concepts manipulés

1) Cubes et Dimensions

Dans cette approche, un cube C est représenté par un schéma en étoile, i.e. un cube N-dimensionnel C est un tuple C = ‹D_1, . . . , D_N, F› où:

- pour i ∈ [1, N], D_i est une table dimension de schéma sch(D_i) = {L_i^0, ….. , $L_i^{d_i}$ }. Pour chaque dimension i ∈ [1, N], chaque attribut L_i^j décrit un niveau d'une hiérarchie, j étant la profondeur de ce niveau.

L'attribut L_i^0 est le niveau le plus fin et il constitue la clé primaire de D_i. Les valeurs du domaine de chaque attribut sont appelées ***membres****.*

- F est la table des faits de schéma sch(F) = {L_1^0,… … , L_1^N, m} où m est l'attribut mesure.

Exemple 4.2 Nous décrivons dans cet exemple, le cube SalesCube de l'exemple 4.1. Ce cube est de schéma sch(SalesCube) = {Time, Product, Location, Measures, Sales} où :

- L'ensemble des dimensions est D = {Time, Product, Location, Measures}. Les schémas de ces quatre tables de dimension, c.-à-d. les schémas de Time, Location, Product et Measures, sont les suivants :
- sch(Time) = {day, month, year, all}.
- sch(Product) = {item, category, all}.
- sch(Location) = {city, region, country, all}.
- sch(Measures) = {name}.

La table des faits du cube, nommée ″Sales″, est de schéma sch(Sales) = {Time, Product, Location, Measures, val} où le domaine de l'attribut ″val″ est l'ensemble de toutes les valeurs de mesure possibles.

Une instance de ce cube est salesCube = {time, product, location, measures, sales}. Un extrait de l'instance time de la table de dimension Time du cube est décrite par la table 4.5 ci-après.

Time	***all***	***Year***	***Month***	***day***
	All	2007	jan/2007	1/jan/2007
	all	2007	jan/2007	2/jan/2007
				. . .
			. . .	. . .
		. . .	. . .	. . .

Tab. 4.5 – Une instance time de la dimension Time du cube SalesCube.

Un exemple d'une instance Sales de la table de faits est donné dans la table 4.6 ci-dessous.

Sales	***Time***	***Product***	***location***	***Measures***	***Val***
	2007	Drink	Paris	quantity	90
	2007	drink	centre	quantity	130
	.	.	.	.	.
	.	.	.	.	.
	.	.	.		.

Tab. 4.6 – Une instance sales de la table de faits du cube SalesCube.

Nous avons décrit un cube en décrivant son schéma et son instance. Un cube de données peut aussi être vu comme un ensemble de cellules où chacune d'elles représente une association entre un membre de chaque dimension et une mesure (ou contenu de cellule).

Un cube de données multidimensionnel est composé de cellules présentant des mesures observées aux croisements des dimensions.

2. Visualisation

Une visualisation présente les données d'un cube à un utilisateur sous forme d'une table croisée. Elle contient les données du cube et est caractérisée par une structure qui permet de préciser la forme de présentation des données du cube à l'affichage.

Nous allons donner dans ce qui suit les définitions du schéma d'une visualisation et de son instance. Pour cela, nous décrivons tout d'abord la structure utilisée pour visualiser les données du cube en définissant formellement le schéma de la structure et son instance.

2. 1. Schéma et instance de visualisation

Afin de décrire avec précision la table croisée utilisée pour afficher une instance donnée d'un cube, on doit avoir un moyen de connaître les caractéristiques exactes de l'affichage. Celles-ci sont définies dans la structure et sont données par : le placement des dimensions sur les différents axes et l'ordre d'apparition des membres de chaque dimension sur un axe donné.

Définition 4.1 (- Schéma de structure). Étant donnée une instance de cube c = {d_1,...., d_N, f } de schéma sch(c) = {D_1, . . . ,D_N, F}, le schéma d'une structure S de c, noté sch(S), est un ensemble de noms de relation défini par :

sch(S) = {Axis, O_{D_1}, . . . , O_{D_N}, } et de leurs schémas tels que :

- sch(Axis) = {numero-axe, ensemble-dimensions} est le schéma de la relation Axis. Une instance de cette relation indique quelles dimensions (la valeur de l'attribut ensemble-dimensions) apparaissent sur quel axe (la valeur de l'attribut numero-axe).
- sch(O_{D_i}) = {membre1,membre2} est le schéma de la relation O_{D_i} . Une instance de cette relation spécifie l'ordre des membres de chaque dimension D_i, i ∈ [1,N], sur un axe. Ceci signifie que pour une dimension donnée, un tuple de cette relation indique qu'une valeur de l'attribut membre1 sera placée avant une valeur de l'attribut membre2 dans la table croisée.

Définition 4. 2 (- Instance de structure). Étant donnée une instance de cube c = {d_1, . . . , d_N, f } de schéma sch(c) = {D_1, . . . ,D_N, F}, une instance de structure S avec K axes, notée s, est un ensemble d'instances de relations :

s = {axis, $<_{D_1}$, . . . , $<_{D_N}$ } tel que :

- axis = {(k, X_k) | k ∈ [1, K + 1]} est une instance de la relation Axis avec :
  - $X_k \subseteq D(c)$ et
  - $X_k \cap X_j = \emptyset$ si $k \neq j$.

- $\prec_{D_i}$ est une instance de relation d'ordre total définie sur $\pi*(d_i)$, i ∈ [1,N].

Le tuple (K + 1, X_{K+1}) de la relation axis indique la tranche du cube qui sera visualisée, et X_{K+1} contient donc les dimensions pour lesquelles un seul membre sera visualisé.

Remarque

La structure peut être :

- Spécifiée dans la requête utilisateur ou
- Définie dans le profil utilisateur sinon
- Calculée.

Nous décrivons une visualisation par un couple composé de données d'un cube et d'une structure d'affichage. Nous définissons le schéma de la visualisation et son instance comme suit.

Définition 4. 3 (- Schéma de visualisation). Étant donnés un cube de données N-dimensionnel C de schéma sch(C) et une structure S de schéma sch(S), le schéma d'une visualisation V des données du cube C, noté sch(V), est un couple de schémas tel que :

sch(V) = (sch(C), sch(S))

Définition 4.4 (- Instance de visualisation). Étant données une instance c du cube de données N-dimensionnel C et une instance s de structure S, une instance d'une visualisation V, notée v, est le couple d'instances défini par :

v = (c, s) où c est une instance de cube C.

Exemple 4.3 Nous décrivons dans cet exemple une structure S pour visualiser une instance de cube salesCube = {time, product, location, measures, sales}. Les données visualisées sont organisées dans une table croisée (figure 4.5). La structure S utilisée est de schéma sch(S) = {Axis, O_{D_1}, O_{D_2}, O_{D_3}, O_{D_4} }

et d'instance s = {axis, $\prec_{D_1}$, $\prec_{D_2}$, $\prec_{D_3}$, $\prec_{D_4}$ } où :

- axis = {(1, {Time}), (2, {Location, Product}), (3, {Measures})}.

Cela signifie que la dimension ″Time″ apparaît sur l'axe 1 et les deux dimensions ″Location″ et ″Product″ sont imbriquées et apparaissent sur l'axe 2. Concernant la dimension ″Measures″, elle n'est placée sur aucun des deux axes de la visualisation car elle identifie la tranche du cube à visualiser, et comporte un seul membre qui est ″quantity″ dans cet exemple.

- Les instances $<_{Time}$, $<_{Product}$, $<_{Location}$ et $<_{Measures}$ des membres des instances ″Time″, ″Product″, ″Location″ et ″Measures″ de dimensions sont :
  - $<_{Time}$= {(2003, 2004), (2004, 2005), (2005, 2006)}*,
  - $<_{Product}$= {(drink, food)} et

- $<_{Location}$= {(Tours, Orleans), (Orleans, Centre)}* .
- $<_{Measures}$= {(quantity, price)}.

où * est le symbole qui désigne la fermeture transitive.

Ce qui signifie, par exemple, que $<_{Time}$ indique que le membre 2003 doit apparaître avant 2004, le membre 2004 doit apparaître avant 2005, etc.

		2003	2004	2005	2006
Tours	drink	77,00	54,00	55,00	33,00
	food	89,00	61,00	30,00	41,00
Orleans	drink	25,00	50,00	49,00	32,00
	food	33,00	44,00	59,00	27,00
Centre	drink	102,00	104,00	104,00	65,00
	food	122,00	105,00	89,00	68,00

Fig. 4.5 – Une instance de visualisation

On note que par la suite, un ensemble d'instances de visualisation de données de cube sera noté V.

3) Contrainte de visualisation

Est une fonction booléenne définie sur un cube et sa structure, elle précise par exemple quelle est la taille de l'écran d'affichage du dispositif, ou plus précisément le nombre maximal de graduations qui peuvent être affichées sur les différents axes d'analyse.

4) Requête OLAP

Dans notre cas le langage utilisé pour l'expression des requêtes est le MDX (Multi Dimensional Expressions). Une requête MDX permet de rechercher des mesures à des niveaux différents de granularité en formulant une expression unique et de spécifier comment afficher la réponse sous forme de table croisée, une instance de visualisation correspond au résultat d'une requête MDX sur une instance du cube. Une instance de visualisation est composée d'un ensemble de faits que l'utilisateur souhaite voir et d'une structure pour l'affichage de la table croisée.

Exemple 4.4 Par exemple, considérons la requête MDX suivante :

```
SELECT {[year].members} ON COLUMNS,
CROSSJOIN ({[Location].Tours, [Location].Orleans, [Location].Centre},
        {[category].food, [category].drink}) ON ROWS
FROM SalesCube
WHERE [Measures].quantity
```

Cette expression MDX recherche la quantité de produits vendus par année et catégorie aux niveaux indiqués de la dimension *Location (city, region).*

De plus, elle indique que les membres des dimensions *Location* et *Product* seront imbriqués et seront affichés en ligne, et que les membres de la dimension *Time* apparaîtront

en colonne. Cela signifie que dans cet exemple, la structure d'affichage du résultat est spécifiée par l'utilisateur en précisant la clause ″ON″ de MDX dans ″SELECT″.

On peut donner maintenant la forme syntaxique générale des requêtes MDX que nous considérons:

SELECT SX_1 *ON AXIS(1),*
. . . ,
SX_K *ON AXIS(K)*
FROM C
WHERE TX;
où :

- SX_k ($k \in [1, K]$) sont des expressions d'ensemble,
- C est un cube de données et
- TX est l'expression d'un tuple.

2. 2. 5. 2. Formulation du problème de personnalisation de visualisation

2. 2. 5. 2. 1. Exemple de motivation

Soit le cube SalesCube = [Time, Location, Product, Measures, Sales] un 4-dimensionnel cube avec :

- sch(Time) = {day, month, year}.
- sch(Product) = {item, category}.
- sch(Location) = {city, region, country}.
- sch(Measures) = {name} with
- adom (name) = {quantity, price}.
- Sch (Sales) = {day, city, item, name, m}.

Supposons que la requête utilisateur sur ce cube soit la suivante : les ventes par catégories de produits, année et région. Cette requête est exprimée en MDX[2] comme suit :

```
SELECT {[City].Tours,[City].Orleans},
{[Year].2003, [Year].2004, [Year].2005,[Year].2006},
{[Category].shoes,[Category].cloth,
Category].food,[Category].drink}
FROM SalesCube
WHERE [Measures].quantity
```

La requête, ci-dessus, dont la réponse est visualisée par le tableau croisé de la figure 4.6, définit les informations suivantes :

[2] *MDX* est un langage de requête créé par *Microsoft* pour manipuler les données multidimensionnelles.

- La clause ″SELECT″ spécifie l'ensemble des membres devant apparaître sur les axes du tableau croisé (tous les membres du niveau ″region″ de la dimension ″Location″, les membres du niveau ″year″ de ″Time″ et les membres du niveau ″category″ de ″Product″.
- La clause ″FROM″ indique que la source de données est le cube ″SalesCube″.
- La clause ″WHERE″ sélectionne le membre ″quantity″ de la dimension ″Measures ″.

Notons que, de plus dans une requête, la syntaxe MDX peut distinguer l'identification des axes d'affichage par ″COLUMNS″ et ″ROWS″ (la structure est définie dans la requête). Ainsi, l'emplacement des informations de la requête sur ces axes est précisé.

Exemple 4.5: Soit la requête *SELECT [region].members ON COLUMNS,*
CROSSJOIN ([year].members, [category].members) ON ROWS
FROM SalesCube
WHERE [Measures].quantity

La réponse à cette requête sera visualisée dans la table de la figure 4.6.

		north	south	east	west
2005	drink	77,00	54,00	55,00	33,00
	food	89,00	61,00	30,00	41,00
	...				
2004	drink	25,00	50,00	49,00	32,00
	food	33,00	44,00	59,00	27,00
	...				
...					

Fig.4.6 – Tableau croisé : ventes par région et par année et catégories de produits.

Dans notre travail, nous nous intéressons particulièrement à la présentation de la réponse. Autrement dit, à la structure utilisée pour la visualisation du résultat.

2. 2. 5. 2. 2. Problèmes

Si le nombre de tuples, réponse à la requête sont très importants alors la réponse complète à la requête peut ne pas être entièrement visualisée à l'écran.

Cependant, si on adapte ce résultat aux souhaits de l'utilisateur, on peut remarquer qu'il existe plusieurs manières de le présenter selon ce que cet utilisateur souhaite voir et analyser. En effet, si par exemple, l'utilisateur de l'exemple précédent est responsable des ventes dans la ville de Tours alors il va s'intéresser au résultat présenté par la figure 4.7. Sinon, si son profil indique qu'il est responsable des ventes récentes de ″drink″ et ″food″ dans les villes de Tours et Orléans, alors il va préférer voir le résultat de la figure 4.8.

city = Tours

	2003	2004	2005	2006
food	72,00	50,00	33,00	89,00
drink	26,00	20,00	25,00	77,00
cloth	56,00	30,00	32,00	60,00
shoes	45,00	50,00	32,00	51,00

Fig. 4.7 – visualisation 1

		2003	2004	2005	2006
Tours	drink	77,00	54,00	55,00	33,00
	food	89,00	61,00	30,00	41,00
Orleans	drink	25,00	50,00	49,00	32,00
	food	33,00	44,00	59,00	27,00

Fig. 4.8 – visualisation 2

Intuitivement notre problème est le suivant :

Etant donnée une requête OLAP et un profil utilisateur, le problème est de trouver une version personnalisée de la requête de telle sorte que sa réponse soit entièrement visualisable sur l'écran tout en répondant aux préférences de l'utilisateur.

Autrement dit, le problème consiste à afficher la meilleure réponse à l'utilisateur, visualisable sur son écran. Notez que le problème est un problème d'optimisation avec contraintes.

2. 2. 5. 2. 3. Formalisation du problème

Input

- Une requête MDX **q**
- Une contrainte de visualisation **v**
- Un profil utilisateur **P**

Output

Un ensemble de sous requêtes Q^* contenant les requêtes les plus intéressantes (répondant au profil P) et satisfaisant la contrainte v.

Formellement, le problème est de trouver l'ensemble de requêtes Q^* défini par :

$$Q^* = \max_{\leq_P} \{q' \in Q_{MDX} \mid q' \sqsubseteq q \wedge v(q') = true\}.$$

Où $\leq_P$ est l'ordre des préférences de l'utilisateur sur les différentes dimensions et membres de cube contenues dans le profil utilisateur.

La solution au problème de personnalisation :

1. Calcule dans une première phase, les sous ensemble les plus intéressants de tuples (C^*) de q qui peut être visualisé en respectant la contrainte v.
2. Détermine à la seconde phase, les structures (S^*) permettant de visualiser l'ensemble le plus intéressant de tuples de q calculé à la première phase.

Note : Le problème de personnalisation peut être vu comme étant un problème d'optimisation avec contraintes : *trouver l'instance de visualisation la plus intéressante du point de vue de l'utilisateur en respectant des contraintes*.

2. 2. 5. 2. 4. Complexité du problème

Notez que plusieurs visualisations peuvent être solution du même problème si les préférences de l'utilisateur sont exprimées par un pré-ordre ou un ordre partiel. De plus, ces visualisations optimales sont MODULO ≡ équivalentes.

Le problème de personnalisation décrit ci dessus est connu pour être NP-complet. Par conséquent, la recherche d'une solution optimale est un processus difficile qui peut prendre beaucoup de temps. En outre, on peut voir qu'une sous classe de ce problème est similaire au problème de sac à dos.

Dans le but de résoudre ce problème dans un temps polynomial, on présente un pré-ordre particulier pour tous les cubes C, et un nombre d'axes égale à deux, ainsi qu'un ensemble particulier de contraintes de visualisation. La complexité de l'algorithme deviendra $O(N\ G_m^2)$ où N est le nombre de dimensions de C et $G_m = max_{\leq p}\{G_1, G_2\}$ et G_m est le nombre maximal de positions que peuvent être visualisées sur les deux axes.

2. 2. 5. 2. 5. Personnalisation de visualisation

Dans les travaux de Bellatreche et al, :

i. La contrainte d'affichage est définie par: La taille maximale de la table croisée qui contiendra la réponse à la requête.

ii. La modélisation des préférences est qualitative.

a) Méthode de personnalisation de [Bellatreche et al, 2005].

Le profil utilisateur dans ce travail est composé des préférences utilisateur définies par un pré-ordre total sur l'ensemble des membres de toutes les dimensions et une contrainte d'affichage. Le problème traité consiste à trouver les sous-cubes les plus intéressants d'un cube C qui peuvent être visualisés. Dénoté par C^* un de ces sous-cubes, est tel que il existe une structure S^* qui permet de le visualiser.

Sachant que si la contrainte de visualisation ne définie pas précisément la structure d'affichage d'un sous-cube optimal C^*, cette dernière doit être trouvée.

Ce qui signifie que ce travail personnalise l'ensemble des faits (contenu) qui sera présenté à l'utilisateur ainsi que la structure (présentation) de ces faits.

On note que plusieurs sous-cubes peuvent être résultat de cette personnalisation puisque la relation entre les membres est un pré-ordre. Ces sous-cubes sont dits modulo équivalents ≡. Intuitivement, on dit qu'ils sont ″également intéressants″. Et plusieurs structures peuvent êtres proposées pour le même sous-cube.

b) Méthode de personnalisation de [Bellatreche et al, 2006].

Exemple 4.6: soit la requête q:

```
SELECT [Location].members ON COLUMS,
CROSSJOIN ([year].members, [category].members) ON ROWS
FROM SalesCube
WHERE [Measures].quantity
```

Dans ce contexte, les requêtes sont exprimées dans le langage MDX. Une requête MDX q est définie par :

1. L'ensemble des références (tuples) des cellules concernées par q, R = ref (q) ;
2. Et la structure utilisée pour afficher cet ensemble de références S(q).

q peut s'écrire q[R, S] tel que :

R= Ref (q) = ref1(q) × ref2(q) × ref3(q) où ref1(q) = adom(Location),
ref2(q) = adom(year) ×adom(category) and
ref3(q) = {quantity}.

S= S(q)= < S1(q), S2(q), S3(q) > with S1(q) = {Location},
S2(q) = {Time, Product} and
S3(q) = {Measures}

A noter que la structure de la réponse est définie dans la requête, la dimension « Location » va apparaitre en colonne (ON COLUMS), et les dimensions « Time » et « Product » imbriquées en ligne (ON ROWS).

Le profil utilisateur contient :

- **Les préférences utilisateur** qui consistent en :
  1. *un ordre partiel strict* sur l'ensemble des dimensions du cube.
  2. un *ordre partiel strict* sur l'ensemble des membres de chaque dimension.

  L'algorithme de personnalisation dans ces travaux prend une requête OLAP comme entrée, et il retourne la (ou les) versions ***personnalisées*** de la requête de l'utilisateur tel que la réponse à cette requête répondra à ses préférences et satisfait la contrainte de visualisation.

A partir des préférences utilisateur sur les membres et les dimensions un ordre sur les références générées par la requête est établi, cet ordre est obtenu comme suit :

Soient t et t′ deux références d'un cube C. Si Δ(t, t′) est l'ensemble des dimensions où t et t' différent, la comparaison de t et t' est basée sur les dimensions les plus importantes de cet ensemble c.à.d. $\max_{<d}(\Delta(t, t'))$. On dit que t est moins important que t' si pour chaque $D_i \in \max_{<d}(\Delta(t, t'))$, nous avons $t(D_i) < t'(D_i)$.

Intuitivement, pour comparer deux tuples t et t' selon un ordre lexicographique $<_P$, on procède comme suit :

1. On commence par calculer l'ensemble des attributs $\Delta(t, t')$ sur lesquels t et t' diffèrent,

2. On calcule ensuite les attributs de $\Delta(t, t')$ qui sont les plus importants selon l'ordre sur les attributs $<_A$, c.-à-d. on calcule $max_{<A}\ \Delta(t, t')$,

3. On compare finalement les composantes de t et t' sur les attributs A_k de $max_{<A}\ \Delta(t, t')$ selon les ordres associés $<A_k$. Si pour tout $A_k \in max_{<A}\ \Delta(t, t')$, on a $t(A_k) <A_k t'(A_k)$, alors on conclut que $t <_P t'$.

Exemple 4.7

Soit le cube SalesCube défini dans l'exemple de motivation, et

P = $[<_d, \{<_{Time}, <_{Product}, <_{Location}, <_{Measures}\}]$ un profil utilisateur défini sur SalesCube par:

- Year $<_d$ Product $<_d$ Location,
- 2002 $<_{Time}$ 2003 $<_{Time}$ 2004 $<_{Time}$ 2005 $<_{Time}$ 2006
- electronics $<_{Product}$ shoes $<_{Product}$ cloth $<_{Product}$ food $<_{Product}$ drink.

a. Centre $<_{Location}$ Orleans $<_{Location}$ Tours.

b. quantity $<_{Measures}$ price.

Tel que $<_d$ est l'ordre des préférences utilisateur défini sur les dimensions et $<_{Time}$, $<_{Product}, <_{Location}, <_{Measures}$ sont les différents ordres définis sur les membres de chaque dimension.

Soit t = <2005, food, Centre, price> et

t'= <2006, drink, Orleans, price> deux références de même schéma

sch(t) = sch(t') = {Time, Product, Location, Measures}.

On a Δ(t, t') = {Time, Product, Location}.

Et $max_{<d}(\Delta(t, t'))$ = {Location} et t(Location) = Centre $<_{Location}$ t'(Location) = Orleans,

Donc : $t \leq_P t'$.

Si l'ordre des préférences sur les dimensions et les membres est partiel alors l'ordre conséquent sur les références sera également partiel. En effet, l'ordre résultant sur les requêtes sera partiel.

Soient deux requêtes q_1 et q_2, q_2 est plus intéressante que q_1 pour un utilisateur donné si son profil indique que les références de cellules de la réponse à q_2 sont préférées à celles de la réponse à q_1.

Méthode de personnalisation:

Utilisant le profil utilisateur, l'algorithme calcule le sous ensemble de références le plus intéressant qui satisfait la contrainte, réécrit la requête suivant cet sous ensemble, ensuite il calcule la structure de la table croisée qui contiendra le résultat (si cette dernière n'est pas spécifiée dans la requête).

L'algorithme retourne une classe d'équivalence de requêtes, ces requêtes sont également intéressantes, et plusieurs structures peuvent êtres trouvées pour chaque requête de cette classe, ceci est dû au faite que l'ordre de préférences introduit par l'utilisateur est partiel.

Pour y remédier, un ordre machine est défini dans le travail de [Mouloudi, 2007] décrit ci-dessous.

c) Travail de [Mouloudi, 2007]

L'ordre de préférence est un ordre total strict, obtenu par combinaison de l'ordre partiel strict introduit dans [Bellatreche et al., 2006] et d'ordres totaux machines.

Définition d'ordre machine (lexicographique)

Afin de définir un ordre total strict sur les préférences utilisateurs (qui sont dans le cas général des ordres partiels), on va supposer l'existence de deux types d'ordres machines :

a. Un ordre machine sur les dimensions du cube considéré,

b. Des ordres sur l'ensemble des membres de chacune des dimensions D_i du cube considéré.

On supposera que ces ordres sont des ordres stricts totaux; en pratique, ils peuvent correspondre à l'ordre lexicographique sur les chaînes de caractères ou à l'ordre de stockage des informations sur le disque.

L'introduction d'ordres totaux machines permettra de fournir à un utilisateur une visualisation personnalisée et une seule, parmi toutes les personnalisations possibles, même si la relation d'ordre induite par ses préférences n'est pas un ordre total strict.

Méthode de personnalisation :

Dans cette méthode, les relations de préférence et contraintes sur les visualisations sont uniquement basées sur leurs ensembles de références et structures.

Après une étape de génération de références, un ensemble de références (ordonné) est obtenu progressivement selon l'ordre de préférence, sur cet ensemble des opérations de test sont effectuées pour vérifier s'il est visualisable. A l'issue de cette méthode nous disposons d'un ensemble maximal de références préférées et visualisable.

La méthode utilise l'ensemble de références pour permettre la personnalisation de deux manières différentes : avant ou après l'accès au cube de données. En effet, nous pouvons

construire la requête personnalisée à partir de cet ensemble de références ou filtrer la réponse de la requête non personnalisée selon cet ensemble.

Notons qu'il a été démontré que la méthode proposée est insensible au fait que l'opérateur de personnalisation soit implanté par transformation de requête ou de la réponse, en terme de contenu de résultat. Cette démonstration ne tient pas compte du coût d'exécution (temps et espace disque).

- ***Comparaison des différentes méthodes***

	[Belletreche et al, 2005]	***[Belletreche et al, 2006]***	***[Mouloudi, 2007]***
Les préférences utilisateur consistent en :	Un pré-ordre total sur l'ensemble des membres de toutes les dimensions.	1) *un ordre partiel strict* sur l'ensemble des dimensions du cube ; 2) un *ordre partiel strict* sur l'ensemble des membres de chaque dimension.	1) *un ordre total strict* sur l'ensemble des dimensions du cube ; 2) un *ordre total strict* sur l'ensemble des membres de chaque dimension.
Personnalisation de quoi :	Cube	Requête	Requête
Quand s'effectue la personnalisation ?	Après exécution de la requête	Avant exécution de la requête	Avant exécution de la requête
L'algorithme proposé prend en entrée :	Un ensemble de membres ordonné.	Un ensemble de références (les tuples)	Un ensemble de références (les tuples)

Tab. 4.7. – Comparaison des différentes méthodes proposées par Bellatreche et al.

Ci-après nous citons les avantages et les inconvénients de la méthode de personnalisation proposée par Bellatreche et al.

- ***Avantages***

1) On n'a pas besoin d'accéder à la table des faits pour personnaliser la requête. En effet, si les tables de dimension et le profil utilisateur résident en mémoire principale, la requête est personnalisée sans accéder à la mémoire secondaire.

Selon sur quoi porte la personnalisation, les méthodes visant à personnaliser le contenu de l'information utilisent les préférences utilisateur sur des requêtes sans avoir l'accès aux données.

- ***Inconvénients***

1) Le résultat de l'application des deux premières méthodes retourne une classe d'équivalence de requêtes (ou bien de cubes pour la première méthode) de même ordre d'importance. Ceci est dû au fait que l'ordre exprimé dans les préférences utilisateur sur les dimensions et les membres est partiel dans la deuxième méthode et pré-ordre sur tout les membres dans la première méthode.

2) Pour y remédier un ordre est imposé par la machine appelé ordre lexicographique, mais cet ordre n'est pas signifiant en termes de préférence.

3) La réponse peut être vide puisque la personnalisation est faite sans accès aux données.

4) Les algorithmes proposés dans les deux dernières méthodes s'appliquent sur les tuples générés par la requête, ce nombre peut être très important, en effet son chargement dans la mémoire centrale pour effectuer le tri peut être très coûteux.

2. 3. Conséquences de la personnalisation sur l'optimisation des requêtes

La résolution de ce problème engendre certains problèmes sur les techniques d'optimisation des requêtes OLAP comme l'Indexation, les vues matérialisées et la fragmentation. [Bellatreche et al, 2005]

Dans ce qui suit nous allons voir l'impact de la personnalisation sur les techniques d'optimisation.

i. Problème de Sélection de Vues Matérialisées

Le problème de sélection de vues matérialisées consiste à choisir un ensemble de vues à matérialiser définies à partir du schéma de la base de données; tel que le coût d'évaluation des requêtes les plus fréquentes soit minimisé et que les vues sélectionnées soient sauvegardées.

Pour cela, les algorithmes considèrent un plan d'accès global à une requête dans lequel les plans d'accès locaux pour les requêtes individuelles sont fusionnés à base des opérations partagées (jointure, union, intersection, etc...) sur les ensembles de données communes. Chaque nœud intermédiaire représente une vue potentielle.

Dans ce cas, tenant compte des profils utilisateur, de nouveaux opérateurs de jointures seront ajoutés, par conséquent le choix des vues à matérialiser est plus provocant, car le nombre de nœuds intermédiaires est augmenté.

ii. Problème de sélection d'index

Compte tenu de la complexité des requêtes décisionnelles et la nécessité d'un temps de réponse court, plusieurs techniques d'indexation ont été développées pour accélérer l'exécution des requêtes. Parmi ces techniques on cite : l'index Bitmap, les B-arbre, l'index de jointure et l'index en étoile de jointure.

La sélection d'un ensemble approprié d'index respectant la contrainte de stockage est un problème difficile, et la personnalisation implique l'ajout de nouvelles conditions de sélection et de jointure à la requête initiale, ces conditions changent les entrées de l'algorithme de sélection d'index et augmentent l'ensemble d'index candidat.

iii. Problème de sélection des partitions

La technique de fragmentation consiste à décomposer le schéma relationnel de l'entrepôt (en étoile ou en flocon de neige) en plusieurs sous-schémas obtenus en se basant sur des sélections sur le schéma initial afin de réduire le nombre des accès nécessaires pour le traitement de certaines requêtes. Cette décomposition est basée sur les opérations de sélection et de projection.

Lors de la personnalisation, de nouvelles conditions de sélections doivent être prises en considération lors du partitionnement ce qui peut influencer sur l'algorithme de partitionnement du fait que la complexité de la plupart des algorithmes de partitionnement dépend du nombre de prédicats de sélection.

3. Conclusion

Dans ce chapitre, nous avons préféré évoquer d'abord les différents concepts liés aux entrepôts, que ce soit au niveau de la modélisation, de la stratégie de conception, de l'exploitation, etc. Cette présentation nous a permis de positionner le contexte général des travaux présentés ainsi que notre travail.

Par la suite, nous avons attaqué le noyau de notre thème qui est la personnalisation, nous avons éclaircit la différence entre personnalisation et recommandation, à quel niveau la personnalisation peut être effectuée : au niveau conception de l'entrepôt ou au niveau présentation (visualisation) des résultats d'une requête OLAP.

Et pour chaque type de personnalisation nous avons fait un rapide survol des différents travaux effectués dessous. Par contre une description détaillée est réservée à la personnalisation des visualisations dans laquelle s'inscrit notre travail. Pour laquelle nous avons présenté la problématique de la personnalisation de la visualisation. Et des solutions qui ont pu être proposées.

Et on s'est focalisé sur la présentation en détail de l'approche de personnalisation des réponses aux requêtes utilisateur interrogeant les données du cube, c.-à-d. des visualisations proposée par [Belletreche et al, 2006]. Dans cette proposition, la visualisation est sous forme d'une table croisée. Après une étape de génération de références, et selon l'ordre de préférence, un ensemble de références est obtenu. Sur cet ensemble seront effectuées des opérations de test pour vérifier s'il est visualisable. A l'issue de cette méthode on dispose d'un ensemble maximal de références préférées et visualisables.

Notre contribution consiste en une amélioration de la méthode de personnalisation proposée dans les travaux de [Mouloudi, 2007], ainsi notre personnalisation portera sur l'appropriation du contenu en satisfaisant la contrainte d'affichage qui est la taille de l'écran.

Dans le chapitre suivant, nous allons présenter notre approche de personnalisation.

DEUXIEME PARTIE

PERSONNALISATION DES VISUALISATIONS DANS LES ENTREPOTS DE DONNEES

Après avoir présenté le problème de la personnalisation dans la première partie de ce mémoire, la deuxième partie est consacrée à la présentation de notre contribution sur la personnalisation des requêtes OLAP, portant sur la visualisation des réponses à ces requêtes.

5 *Personnalisation de visualisation*

Dans ce chapitre nous allons développer notre contribution sur la personnalisation des visualisations des réponses aux requêtes OLAP. Nous commençons par une description globale de notre approche qui est venue palier aux insuffisances de la méthode de personnalisation proposée par [Mouloudi et al, 2007] décrites dans le chapitre précédent. Nous formalisons par la suite le problème de personnalisation traité. Puis nous détaillons notre approche en donnant les algorithmes proposés.

1. Notre contribution

Comme nous l'avons vu précédemment, le travail de Mouloudi sur la personnalisation de visualisation se base sur l'ordonnancement des références de cellules (tuples) générées par la requête, ce nombre de tuples peut être très important, vu que la charge informationnelle manipulée dans les entrepôts de données est importante. En effet, l'ordonnancement de cet ensemble de tuples peut engendrer un temps d'exécution long. Cependant vu que le dispositif d'affichage est un téléphone portable ou un PDA, le nombre de cellules à afficher est très petit et l'objectif de ce travail est d'éviter à l'utilisateur de naviguer longuement pour avoir l'information pertinente. En effet, on doit réduire au minimum le nombre de tuples manipulés.

Notre contribution est venue palier au problème d'ordonnancement des références de cellules, problème qui peut surgir lorsque le nombre de références est important, et c'est souvent le cas dans les entrepôts de données.

Notre approche, exploite le fait que l'écran recevant le résultat de la requête est petit, cet écran est vu en tant que nombre de cellules. En conséquence, notre approche se base essentiellement sur la décomposition du nombre entier représentant cette taille de l'écran (contrainte d'affichage) en facteurs de *nombres premiers*.

Ces facteurs vont nous servir à préciser le nombre de membres de chaque dimension à afficher sur chaque axe d'analyse.

Justification du choix de décomposition en facteurs de nombres premiers :

Le problème que nous traitons est un problème d'optimisation de l'affichage du résultat d'une requête, et ce car le nombre de cellules visualisables est limité.

Notre problème consiste à placer les dimensions d_i contenant des membres m_i^j i = 1..N, j=1.. k sur les k axes d'analyse en respectant une contrainte d'affichage : nombre maximal de cellules à afficher.

En effet, le problème devient : combien de membres de chaque dimension peut-on afficher sur chaque axe d'analyse?

Sachant que nous nous disposons que du nombre maximal de cellules (qui est un nombre entier), et nous cherchons à : à partir des ce nombre avoir un ensemble maximal de nombres :

- Chacun de ces nombres sera affecté à une dimension, et ce nombre consistera en le nombre de membres de chaque dimension à afficher ;
- Et de sorte à ce que le produit de ces nombres donne le nombre initial (représentant la contrainte d'affichage).

Et comme nous savons, à partir du théorème fondamental de l'arithmétique que tout nombre entier supérieur ou égal à 2 peut être exprimé en un unique produit de nombres premiers, et que cette décomposition est la plus *élémentaire*, elle donne un maximum de facteurs. Nous avons opté pour cette décomposition car : La décomposition en facteurs de nombres premiers, nous donne un ensemble de base, contenant un maximum de facteurs.

Avec cette décomposition on est descendu au niveau le plus élémentaire. En conséquence le nombre de facteurs peut être supérieur au nombre de dimensions de la requête, comme il peut être inférieur ou bien égal. Ces cas seront traités dans la section 4.2 suivant les propositions qu'on a suggéré.

En effet, toute autre alternative de recherche de nombres dont le produit donne l'entier représentant la contrainte d'affichage, peut être amenée à cette décomposition en facteurs premiers.

2. Formalisation du problème

Entrées :

- *Requête MDX Q;*
- *Préférences utilisateur sur les dimensions ($<_d$) et sur les membres de chaque dimension ($<_{d_i}^{m}$);*
- *Contrainte d'affichage : nombres maximaux de cellules à afficher sur ligne et sur colonne.*

Sortie :

- *Requête personnalisée, Q^*;*

Les préférences utilisateur sur les dimensions et les membres de chaque dimension sont exprimées par un ordre partiel, cet ordre est totalisé en introduisant un ordre machine.

Une contrainte est une fonction booléenne sur l'ensemble des visualisations V. Elle est utilisée pour exprimer des critères que les visualisations doivent satisfaire, dans notre cas qu'elle doit comporter un nombre maximal de cellules.

Le problème consiste à réécrire la requête de manière à ce que la réponse à cette requête soit la meilleure par rapport aux préférences de l'utilisateur et que cette réponse soit (entièrement) visualisable sur son écran, c.à.d. que la contrainte de visualisation soit satisfaite.

Formellement, le problème consiste à trouver la requête Q^* la plus intéressante telle que sa visualisation *V* (la table croisée contenant la réponse à cette requête) satisfait la contrainte μ_V, c.-à-d. on cherche la requête Q^* définie par :

$$Q^* = \max_{<_p}\{q' \in Q_{MDX} \mid q' \sqsubseteq q \wedge \mu_V(q') = true\}.$$

$<_p$: représente les préférences de l'utilisateur.

3. Définition des concepts utilisés

Définition 5.1 -Contrainte de visualisation. *Étant donné un ensemble d'instances de visualisations* V, *une contrainte de visualisation* μ_V *sur* V *est une fonction booléenne définie sur* V. *Étant donnée une instance de visualisation* $v \in$ V, *on dit que v satisfait la contrainte* μ_V *si :*

$$\mu_V(v) = true.$$

Définition 5.2 - Préférences utilisateur de contenu. *Étant donnée une instance de cube* $c = \{d_1, \ldots, d_N, f\}$ *de schéma* $sch(C) = \{D_1, \ldots, D_N, F\}$, *des préférences utilisateur de contenu sur c sont définies par un tuple :*

$$U = \langle <_D^u , \{<_{D_1}^u , \ldots , <_{D_N}^u \}$$

où :

- $<_D^u$ est un ordre strict sur l'ensemble des dimensions $D = \{D_1, \ldots, D_N\}$ du cube. Étant données deux dimensions D_i et D_j de D, on dit que D_j est préférée à D_i si $Di <_D^u D_j$.
- pour tout $i \in [1,N]$, $<_{D_i}^u$ est un ordre strict sur l'ensemble des membres de chaque dimension $D_i \in$ D (c.-à-d., un ordre partiel sur $\pi^*(d_i)$, $i \in [1,N]$). Étant donnés deux membres m et m′ de D_i, on dit que m est moins intéressant que m′ si $m <_{D_i}^u m'$.

Exemple 5.1 Soit le cube SalesCube = [Time, Location, Product, Measures, Sales] un 4-dimensionnel cube avec :

- sch(Time) = {day, month, year}.
- sch(Product) = {item, category}.
- sch(Location) = {city, region, country}.
- sch(Measures) = {name} with
- adom (name) = {quantity, price}.
- Sch (Sales) = {day, city, item, name, m}.

Nous considérons dans cet exemple des préférences utilisateur de contenu exprimées sur le cube SalesCube comme suit :

– L'utilisateur préfère la dimension ″Location″ aux deux dimensions ″Time″ et ″Product″, c.à.d.:

- Time $\preceq^{u}_{D}$ Location.
- Product $\preceq^{u}_{D}$ Location

Cela signifie que les dimensions Time et Product sont moins importantes que Location. Notons que Time, Product et Measures ne sont pas comparables selon $\preceq^{u}_{D}$.

– L'utilisateur spécifie ses préférences pour chacune des dimensions comme suit :

- (2003 $\preceq^{u}_{\text{Time}}$ 2004 $\preceq^{u}_{\text{Time}}$ 2005 $\preceq^{u}_{\text{Time}}$ 2006) et (2005 $\preceq^{u}_{\text{Time}}$ 2007).
- electronics $\preceq^{u}_{\text{Product}}$ shoes $\preceq^{u}_{\text{Product}}$ cloth $\preceq^{u}_{\text{Product}}$ food $\preceq^{u}_{\text{Product}}$ drink.
- (Centre $\preceq^{u}_{\text{Location}}$ Tours) et (Centre $\preceq^{u}_{\text{Location}}$ Orleans).
- price $\preceq^{u}_{\text{Measures}}$ quantity.

Notons que 2006 et 2007 ne sont pas comparables selon $\preceq^{u}_{\text{Time}}$, de même que Tours et Orleans selon $\preceq^{u}_{\text{Location}}$.

On peut finalement représenter les préférences utilisateur de contenu par le tuple

$U = \langle \preceq^{u}_{D} , \{\preceq^{u}_{\text{Time}} , \preceq^{u}_{\text{Product}} , \preceq^{u}_{\text{Location}}, \preceq^{u}_{\text{Measures}} \}$

➢ ***Introduction d'ordres machines***

Afin de définir un ordre total strict sur les grilles d'analyse à partir de préférences utilisateurs (qui sont dans le cas général des ordres partiels), on va supposer l'existence de deux types d'ordres machines. Étant donné une instance de cube $c = \{d_1, \ldots, d_N, f\}$ de schéma $sch(c) = \{D_1, \ldots, D_N, F\}$, ces deux ordres sont :

- Un ordre machine noté $\preceq^{m}_{D}$ sur les dimensions D = $\{D_1, \ldots, D_N\}$ du cube considéré,

- Des ordres machine notés $\prec_{D_i}^{m}$ sur l'ensemble des membres $\pi^*(d_i)$ de chacune des dimensions D_i du cube considéré.

On supposera que ces ordres sont des ordres stricts totaux; en pratique, ils peuvent correspondre à l'ordre lexicographique sur les chaînes de caractères ou à l'ordre de stockage des informations sur le disque. Par la suite, on note T le tuple défini par :

$\mathbf{T} = \langle \prec_{\mathbf{D}}^{m} , \{\prec_{\mathbf{D}_1}^{m} , \ldots , \prec_{\mathbf{D}_N}^{m} \}$

Exemple 5.2 Considérons les dimensions et les membres de l'exemple 5.1. Un ordre machine sur ces dimensions et ces membres est comme suit :

– sur les dimensions :

Time $\prec_{\mathbf{D}}^{m}$ Product $\prec_{\mathbf{D}}^{m}$ Location.

– sur les membres :

- 2003 $\prec_{\mathbf{Time}}^{m}$ 2004 $\prec_{\mathbf{Time}}^{m}$ 2005 $\prec_{\mathbf{Time}}^{m}$ 2006 $\prec_{\mathbf{Time}}^{m}$ 2007.
- shoes $\prec_{\mathbf{Product}}^{m}$ f ood $\prec_{\mathbf{Product}}^{m}$ electronics $\prec_{\mathbf{Product}}^{m}$ drink $\prec_{\mathbf{Product}}^{m}$ cloth.
- Tours $\prec_{\mathbf{Location}}^{m}$ Orleans $\prec_{\mathbf{Location}}^{m}$ Centre.
- quantity $\prec_{\mathbf{Measures}}^{m}$ price.

On peut représenter ces ordres machines par le tuple

$\mathbf{T} = \langle \prec_{\mathbf{D}}^{u} , \{\prec_{\mathbf{Time}}^{m} , \prec_{\mathbf{Product}}^{m} , \prec_{\mathbf{Location}}^{m} , \prec_{\mathbf{Measures}}^{m} \}$

Notons que les dimensions et les membres qui ne sont pas comparables selon l'ordre utilisateur, sont tous comparables selon l'ordre machine.

Á partir de préférences utilisateurs de contenu $\mathbf{U} = \langle \prec_{\mathbf{D}}^{u} , \{\prec_{\mathbf{D}_1}^{u} , \ldots , \prec_{\mathbf{D}_N}^{u} \}$ et d'ordres machines $\mathbf{T} = \langle \prec_{\mathbf{D}}^{m} , \{\prec_{\mathbf{D}_1}^{m} , \ldots , \prec_{\mathbf{D}_N}^{m} \}$ définis sur *c*, on peut maintenant introduire successivement les ordres définis comme suit :

– $\prec_{\mathbf{D}} = \prec_{\mathbf{D}}^{u} \lhd \prec_{\mathbf{D}}^{m}$. Comme $\prec_{\mathbf{D}}^{m}$ est un ordre total strict sur l'ensemble des dimensions D, cela implique que l'ordre $\prec_{\mathbf{D}}$ obtenu par composition prioritaire est aussi un ordre total strict sur D.

– Pour tout $i \in [1,N]$, $\prec_{\mathbf{D_i}} = \prec_{\mathbf{D_i}}^{u} \lhd \prec_{\mathbf{D_i}}^{m}$. Comme pour tout $i \in [1,N]$, $\prec_{\mathbf{D_i}}^{m}$ est un ordre total strict sur l'ensemble des membres $\pi^*(d_i)$ de D_i, cela implique que l'ordre $\prec_{\mathbf{D_i}}$ obtenu par composition prioritaire l'est aussi.

Dans ce qui suit, nous décrivons notre contribution au problème de personnalisation de visualisation.

- ***Contraintes de visualisation***

Nous introduisons dans cette section des classes particulières de contraintes de visualisation. Ces contraintes sont de deux types :

– Les premières, dites *contraintes utilisateurs*, sont liées à un utilisateur ; elles précisent comment un utilisateur veut visualiser ses données, par exemple en précisant sur quel axe il souhaite voir telle dimension.

– Les secondes, dites *contraintes de dispositif*, sont liées à un dispositif d'affichage particulier (*PDA*, téléphone mobile, etc.). Elles précisent par exemple quelle est la taille de l'écran d'affichage du dispositif, ou plus précisément le nombre maximal de graduations qui peuvent être affichées sur les différents axes d'analyse.

Notons que, dans notre travail, nous considérons que l'utilisateur exprime la première contrainte dans sa requête MDX, dans les clauses *"ON ROWS"* et *"ON COLUMNS"*, sachant qu'avec le MDX l'utilisateur n'est pas obligé de remplir les clauses *"ON ROWS"* et *"ON COLUMNS"* et dans ce cas le serveur OLAP calcul une structure pour le résultat.

Par contre, le deuxième type de contraintes dit de dispositif est stocké dans son profil. Et est définie formellement comme suit :

Définition 5.3 - Contrainte de dispositif. Soit c une instance de cube de données de schéma sch(c) = {D_1, . . . ,D_N, F}. Étant donné un tuple M = ⟨M_1, . . . ,M_K⟩ de K entiers strictement positifs représentants le nombre d'axes d'analyse, la contrainte de dispositif, notée μ_M, est définie pour toute table croisée T : pour tout k ∈ [1, K], NbGrad(T, X_k) ≤ M_k si (k,X_k) ∈ axis.
NbGrad : est le nombre de graduations qui peuvent êtres visualisées sur l'axe X_k.

4. Algorithmes de personnalisation

4. 1. Exemple de motivation

Exemple 5.3 : Soit le cube SalesCube, et

P = [$<_d$, {$<_{Time}$, $<_{Product}$,$<_{Location}$,$<_{Measures}$}] un profil utilisateur défini sur SalesCube par:

- Year $<_d$ Product $<_d$ Location,
- 2002 $<_{Time}$ 2003 $<_{Time}$ 2004 $<_{Time}$ 2005 $<_{Time}$ 2006
- electronics $<_{Product}$ shoes $<_{Product}$ cloth $<_{Product}$ food $<_{Product}$ drink.

a. Centre $<_{Location}$ Orleans $<_{Location}$ Tours.

b. quantity $<_{Measures}$ price.

- $<_d$: est un ordre total sur les dimensions.
- $<_{Product}$, $<_{Location}$, $<_{Time}$, $<_{Measures}$: sont les différents ordres totaux sur les membres de chaque dimension.

La contrainte d'affichage: G=24=4×6;

Et soit la requête Q=

```
SELECT {[year].2003, [year].2004, [year].2005, [year].2006} ON COLUMNS,
CROSSJOIN ({[Location].Tours, [Location].Orleans, [Location].Centre },
        {[product].food, [product].drink, [product].shoes, [product].cloth }) ON ROWS
FROM SalesCube
WHERE [Measures].quantity
```

Etant donnée la contrainte G= 4×6 c'est-à-dire que le nombre maximal de tuples visualisable est 24 alors on peut visualiser sur ligne **(i)** trois membres de la dimension « *product*» et deux membres de la dimension « *Location* » ou bien **(ii)** trois membres de la dimension « *Location*», deux membres de la dimension « *product*», mais pour l'utilisateur quelle est la meilleure proposition (i) ou (ii) ?

Est-ce qu'il souhaite faire une analyse sur un maximum de membres de sa dimension préférée ? Ou bien c'est le contraire c.à.d. analyser un minimum de membres de sa dimension préférée sur un maximum de membres d'autres dimensions.

Proposition 5.1: Si $D_1 <_P D_2$ alors card (D_1) > card (D_2) càd que le cardinal d'une dimension (le nombre de membres qu'elle contient) est inversement proportionnel à son ordre de préférence ($<_P$).

Intuitivement, si par exemple un utilisateur s'intéresse plus à la dimension « product » qu'à la dimension « Location » alors il va préférer voir

- ✓ les ventes de 2 produits dans 3 villes que
- ✓ les ventes de 3 produits dans 2 villes

En effet, d'après la proposition pour répondre aux préférences de l'utilisateur de l'exemple de motivation, c'est le premier cas qui est porteur car Product $<_P$ Location et card (Product) >. card (Location)

Se basant sur cette proposition, notre algorithme se charge d'affecter les facteurs de la décomposition du nombre représentant la taille de l'écran, aux cardinaux des dimensions impliquées dans la requête de l'utilisateur suivant ses préférences.

4. 2. Calcul de la requête personnalisée

Dans cette section, étant donnée une requête MDX q (contenant obligatoirement dans notre cas les clauses *« on rows »* et *« on columns »*, spécifiant le placement des différentes dimensions sur les deux axes) et une contrainte d'affichage G contenu dans le profil utilisateur. Ce dernier contient aussi ses préférences sur les dimensions et sur les membres de chaque dimension.

Le calcul de la requête personnalisée peut être décomposé en quatre étapes :

1. A la première phase, on ordonne l'ensemble des membres de chaque dimension suivant les préférences utilisateur (fonction *OrdreMember*);
2. Ensuite on décompose le nombre représentant la taille maximale de l'écran en facteurs de nombres premiers (fonction *décomposition*);
3. Puis, on affecte les facteurs obtenus à l'étape deux aux cardinaux des ensembles ordonnés à la première phase de manière inversement proportionnelle aux préférences de l'utilisateur sur les dimensions (suivant la proposition 5.1) (fonction *Affect : étapes 2, 3*).
4. Ensuite on extrait de chaque dimension les k_i meilleurs membres de chaque dimension d_i (k_i c'est le cardinal affecté à chaque dimension) (fonction *Affect : étapes 5,6*). Et une requête personnalisée est générée (fonction *Affect : étape 11*).

1) Fonction OrdreMember

Function *OrderMem*

Input:

- Des ensembles de membres de chaque dimension $d_i = \{m_i^j\}$ i = 1..N, j= 1 .. k_i (de la requête)
- Un ordre total sur les membres de chaque dimension $\prec_{d_i}$ (du profil)

Output: des ensembles ordonnés de membres

---

1. **For** *i* = 1 to *N*.
   2. **Let** $D_i^{ord} = \{m_i^*\}$ où $m_i^* \in max_{\prec_{D_i}}$
   3. **let** $D_i = D_i - \{m_i^*\}$
4. **End for**

// Initialisation de chaque ensemble D_i^{ord} à son meilleur membre m_i^* selon les préférences de l'utilisateur.

5. **For** *i* = 1 to N.
   6. J:= 1
   7. **While** ($D_i \neq \emptyset$) **do**
      8. **Let** $m_i^* = m_i^j \in max_{\prec_{D_i}}$
      9. **Let** $D_i^{ord} = D_i^{ord} \cup \{m_i^*\}$
      10. **let** $D_i = D_i - \{m_i^*\}$
      11. **j:=j+1**
   12. **End while**
13. **End for**

14. **Return** D_i^{ord}

Cette fonction ordonne les membres de chaque dimension selon les préférences de l'utilisateur.

En appliquant cette fonction sur notre exemple de motivation, on aura en sortie les ensembles ordonnés suivants :

Year^{ord}= ‹ 2006, 2005, 2004, 2003 ›

Product^{ord} = ‹ drink, food, cloth, shoes ›

Location^{ord} = ‹ Tours, Orleans, Centre›

2) Fonction décomposition

On ne connaît pas exactement quelles classes de complexité contiennent le problème de la décomposition d'un entier en produit de facteurs premiers, mais dans notre cas la complexité du problème ne se pose pas car notre nombre à décomposer est petit puisqu'il représente la taille maximale de l'écran (nombre de cellules visualisables). Cette fonction retourne un ensemble de facteurs, dont le cardinal est appelé dans ce qui suit nb_{fact}.

Description

Nous pouvons décrire un algorithme récursif pour accomplir de telles factorisations : soit un nombre donné *n.*

- si *n* est premier, alors la factorisation s'arrête ici.
- si *n* est composé, diviser *n* par le premier nombre premier p_1. S'il est divisé sans reste, reprendre avec la valeur n/p_1. Ajouter p_1 à la liste des facteurs obtenus pour n/p_1 pour avoir une factorisation pour n. S'il est divisé avec reste, diviser *n* par le nombre premier suivant p_2, et ainsi de suite.

Notez que nous avons besoin de tester seulement les nombres premiers p_i tels que $p_i \leq n$.

L'algorithme décrit ci-dessus marche bien pour les petits *n*, mais devient impraticable dès que *n* devient plus grand. Ce n'est pas le cas pour nous, car notre n à décomposer représente la taille maximale de l'écran qui ne peut pas excéder un certain seuil.

La décomposition des nombres représentants la contrainte d'affichage de notre exemple de motivation G= 4×6 donne G= (2×2)×(2×3)

3) Fonction affectation

Soit nb_{dim} *le nombre de dimension impliqué dans la requête.*

Plusieurs cas peuvent se présenter :

Cas N° 1 : *Si* $nb_{fact} = nb_{dim}$ *:*

Affecter le plus petit facteur à la dimension la plus intéressante, le facteur suivant à la dimension suivante selon la préférence de l'utilisateur, etc… de manière inversement proportionnelle.

Function A*ffect*

Input

- Ensembles de membres ordonnés de chaque dimension $d_i = \{m_i^j \quad j = 1 \ldots k\} \; i = 1 \ldots N$
- Ensemble de facteurs premiers F dont le produit représente le nombre maximal de cellules (contraint G)

Output

- Requête personnalisée

---

1. ***for*** $i = 1$ *to* N *do*
2. *Let* $f_<$ = *le plus petit des facteurs de F*
3. *Let* $d_i^* = d_i^* \cup max(D_i)$ *// la dimension la plus préférée*
   4. ***for*** *j=1 to* $f_<$ *do*
   5. $d_i^* = d_i^* \cup \{m_i^j\}$ *// extraire les* $f_<$ *membres*
   6. ***end for***
7. $F = F - \{f_<\}$
8. $D = D - d_i^*$
9. ***end for***
10. ***Let*** $Q^* = \prod_{i=1}^{N} d_i$
11. ***Return*** (Q^*)

Cas N° 2 : *Si* $nb_{fact} < nb_{dim}$ *:*

Proposition N° 5.2 :

Compléter les facteurs par des 1 jusqu'à avoir un nombre de facteurs égal au nombre de dimensions, ensuite affecter les facteurs aux dimensions comme fait dans le cas N°1.

1. ***for*** $i = 1$ *to* $nb_{\text{dim}} - nb_{\text{fact}}$ *do*
2. $F = F \cup \{1\}$
3. *Appliquer* ***Affect***

Cas particuliers

Supposons qu'on a G = 9 = 3×3 et le nombre de dimensions = 3 dans ce cas il vaut mieux travailler avec une taille = 8 c.à.d. laisser une cellule vide que affecter un cardinal de 3 membres à 2 dimensions et 1 membre pour la troisième et adopter 8 au lieu de 9 permettra d'avoir 2 membres dans chaque dimension.

Si le nombre de dimensions est supérieur au nombre de facteurs alors réduire le nombre représentant la taille de l'écran au nombre qui lui est plus proche se décomposant en nombre de facteurs égal ou presque égal au nombre de dimensions.

Par exemple : si G = 34 et nombre de dimensions = 5 alors vaut mieux travailler avec G'= 32= 2×2×2×2×2 , que travailler avec 34=2×17.

Cas N° 3 : *Si $nb_{fact} > nb_{dim}$:*

Regrouper les facteurs de manière à avoir un nombre de facteurs égal au nombre de dimensions. Par exemple si G= 12= 2×2×3 et nb_{dim}= *2* alors *il faut regrouper les facteurs deux à deux et on aura deux cas possibles 12= 4×3 et 12=6×2 laquelle des décompositions adopter ?*

***Proposition 5.2* :** regrouper les facteurs de manière à avoir des facteurs rapprochés, pour permettre l'analyse du maximum de membres de chaque dimension, et ce en multipliant les plus petits nombres deux à deux.

Par exemple la décomposition *12= 4×3* sera préférée à la décomposition *12=6×2, si les dimensions sur lesquelles porte l'analyse sont Product et Location, l'utilisateur préférera voir les ventes de 3 produits dans 4 villes que voir les ventes de 2 produits dans 6 villes.*

c.à.d. permettre à l'utilisateur l'analyse sur un nombre maximum de membres de chaque dimension.

1. *While $nb_{fact} > nb_{dim}$ do*
2. $F = F - \{f_i, f_j\} \cup \{f_i \times f_j\}$ *tel que f_i et f_j sont les plus petits facteurs de F*
3. *Appliquer* ***Affect***

5. L'impact de notre approche personnalisation sur les techniques d'optimisation des requêtes

Dans la section 2.3 du chapitre précédent, nous avons présenté les conséquences de la personnalisation sur les techniques d'optimisation, rappelons qu'il a été dit que lors de la personnalisation, de nouvelles conditions de sélections et de jointures prises du profil utilisateur doivent être prises en considération, en conséquence le nombre de sélections augmente ce qui complique le problème de l'optimisation des requêtes quelque soit la technique utilisée.

Signalons que le nombre de sélections augmente car, le profil de l'utilisateur est *intégré* dans la requête initiale.

Par contre dans notre algorithme de personnalisation, on extrait (filtre) des sélections de la requête initiale celles qui répondent le plus aux préférences de l'utilisateur contenues dans son profil. En effet, le nombre de sélections de la requête *diminue*. Ce qui aura un impacte positif sur les techniques d'optimisation. Ceci sera prouvé dans nos tests d'évaluation montrés dans le chapitre réalisation.

6. Conclusion

Nous avons proposé une approche de personnalisation des réponses aux requêtes utilisateur interrogeant les données du cube, c.-à-d. des visualisations. Dans notre proposition, la visualisation est sous forme d'une table croisée composée de deux axes : ligne et colonne. Après extraction des dimensions et membres de chaque dimension de la requête initiale, un ordre est établi sur l'ensemble des dimensions ainsi que sur chacun des ensembles de membres selon les préférences utilisateur.

Le nombre entier représentant la contrainte d'affichage (taille maximale de la table croisée) est décomposé afin d'affecter chaque facteur obtenu à une dimension, ce facteur représente le nombre de membres d'une dimension affichable. Ainsi la requête initiale sera écrite en fonction de ces membres.

Notons que, dans cette approche la personnalisation de la requête consiste en une optimisation implicite du coût d'exécution de cette requête, car réduire le nombre de membres manipulé par la requête implique réduire le nombre d'accès à la table des faits. En outre, notre algorithme de personnalisation réduit le nombre de sélections de la requête, ce qui influe positivement sur les techniques d'optimisation des requêtes.

Dans le chapitre suivant, nous montrons comment l'approche de personnalisation que nous avons proposé dans ce chapitre peut être utilisée dans un domaine d'application.

Nous présentons également les résultats de l'implémentation de cette méthode en utilisant l'entrepôt de données *"Foodmart"*.

TROISIEME PARTIE

VALIDATION DE NOTRE CONTRIBUTION

La dernière partie de ce travail traite la réalisation effectuée présentée dans le chapitre 6. Nous présentons le prototype Mob_OLAP pour les applications mobiles, en décrivant tout d'abord les outils et l'environnement avec lesquels ce système a été réalisé. Puis, nous décrivons son architecture, ainsi que son fonctionnement. Nous donnons ensuite les résultats des expérimentations.

6 Réalisation

La performance en terme de taille de visualisations du résultat d'exécution d'une requête OLAP sur un dispositif mobile est un problème clé, vu que ces dispositifs sont caractérisés par leur petite taille d'écran et que la réponse à une requête OLAP interrogeant des quantités de données importantes est aussi d'une taille importante. En effet, l'utilisateur doit naviguer longuement afin d'obtenir l'information pertinente qu'il recherche.

Pour palier à ce problème, nous avons présenté dans le chapitre précédent notre approche de personnalisation des requêtes décisionnelles prenant en compte les profils des utilisateurs. Cette approche est concrétisée par le système baptisé « *Mob_Olap* » que nous allons présenter dans le présent chapitre.

La base même de la conduite d'un projet informatique quelque soit sa nature repose sur quatre questions : ”Quoi faire ?”, ”Avec quels moyens le faire ?”, ”Comment le faire ?” et ”quel est le taux de réussite du système réalisé ? ”. L'objectif du présent chapitre est de répondre à ces interrogations pour déterminer l'approche à adopter pour mettre en œuvre notre projet.

A la première question ”Quoi faire ?”, nous avons répondu dans le paragraphe précédent. Pour la réalisation de notre système, on a utilisé une boite à outils. Cette dernière sera présentée dans la première section de ce chapitre ceci pour répondre à la question ”Avec quels moyens le faire ?”. La deuxième section détaillera le fonctionnement du système répondant ainsi à la question ”Comment ?”.

Et enfin, pour valider notre solution une série de tests d'évaluation sera effectuée sur ce système. Cette expérimentation est illustrée dans la troisième section de ce chapitre.

1. Architecture du système

Notre implémentation de personnalisation de requêtes OLAP est accessible à partir de n'importe quel terminal (un téléphone portable, un PDA,…), à toute heure et en tout lieu. C'est le paradigme *"Anywhere, Any time"*. Ces derniers sont très loin d'atteindre les vitesses et capacités de traitement des ordinateurs de bureau. Les téléphones disposent de très peu de mémoire et ne bénéficient que de quelques lignes affichables à l'écran ainsi que d'un clavier numérique peu propice à la saisie de texte. C'est pour cette raison que l'architecture qui s'impose pour la mise en œuvre de notre système est l'architecture client/serveur.

Le client/serveur est avant tout une technique de dialogue [Gallo, 2002][Colonna, 2002][Web 5] entre deux processus, qui s'appuie sur l'envoi de requêtes et de réponses en sens inverse. Ces deux processus ne sont pas identiques mais forment un système coopératif (figure 6.1).

Plusieurs serveurs sont utilisés dans l'architecture de notre système, chaque serveur est nommé selon le service qu'il propose à l'utilisateur: serveur web, serveur d'application, serveur de données et serveur OLAP (le rôle de chacun de ces services sera donné dans la section suivante ainsi qu'une brève description de celui retenu pour la réalisation de notre système). L'architecture résultante est appelée multi tiers (figure 6.2).

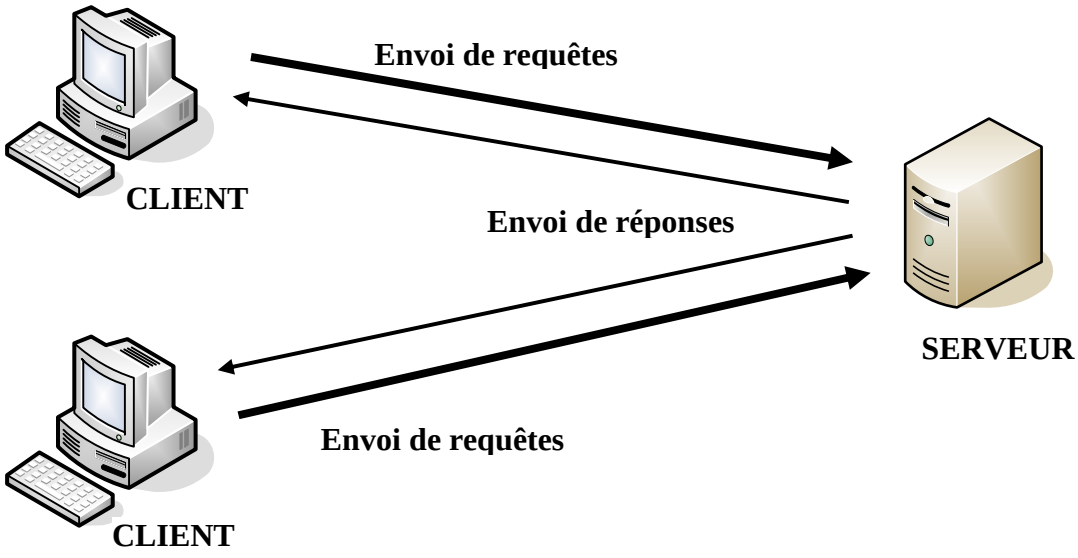

Fig. 6.1. – Le dialogue client/serveur

1. 1. Serveurs d'application

Un serveur d'application est un serveur (ordinateur), sur lequel sont installées des applications utilisées par les particuliers. Elles sont accédées par le réseau. Ces applications, qui étaient souvent disposées sur différents postes, sont aujourd'hui de plus en plus centralisées sur des serveurs d'application. Ces serveurs sont de larges systèmes contenant les différentes applications de l'entreprise.

Dans une infrastructure régulière, on trouve plusieurs serveurs d'applications, mais il n'est pas impossible qu'il n'y en a qu'un seul, sur lequel toutes les applications seraient installées. Les applications sont chargées sur le serveur et leur résultat est affiché sur les écrans des terminaux utilisés par les clients. [Web 2]

1. 2. Serveurs Web

Le World Wide Web est le modèle client/serveur qui consiste en des clients, légers, portables et universels qui communiquent avec de très gros serveurs. Dans la concrétisation la plus simple, un serveur Web renvoie des documents lorsque le client les demande par leur nom. Clients et serveurs communiquent par un protocole appelé HTTP. Celui-ci définit un jeu de commandes simples, où les paramètres sont transmis comme des chaînes de caractères sans typage particulier.

Dans notre cas, nous avons choisi le couple Apache / Tomcat qui assure simultanément les deux services précédemment décrits.

Présentation du couple Apache / Tomcat

Tomcat est un serveur d'application Java permettant d'exécuter des servlets et des pages serveur Java (JSP). Il est développé sous licence open-source par la fondation Apache. Il peut être utilisé ou couplé avec un serveur Web (dont Apache), et porté sur n'importe quel système sur lequel une machine virtuelle Java est installée. [Web 3]

Notre choix de ce couple est justifié par ses principaux avantages qui tiennent de son mode de distribution et à son mode de licence. Il s'agit d'un produit gratuit, ouvert, disponible aussi bien sous forme binaire exécutable que sous forme de source. Les avantages de ce type de distribution sont énormes. L'avantage principal n'est pas la gratuité. Le Serveur HTTP est de plus libre, et offre une totale indépendance de l'utilisateur pour la maintenance et le développement.

Le module Tomcat

Le conteneur de servlets choisi est le moteur Tomcat développé par la fondation Apache [House, 2005] [Axis, 2005]. Le dialogue entre le moteur de servlet et le serveur Web

s'effectue à l'aide d'un module logiciel appelé connecteur. Tomcat peut fonctionner sur d'autres serveurs Web mais seul le couple Tomcat/Apache a été testé. Le module Tomcat du serveur Apache a été développé à partir des sources de Sun Microsystems. Il représente une implémentation de référence pour les servlets. Tomcat peut fonctionner seul, mais cela n'est pas une solution efficace. En exploitation il est préférable d'associer Tomcat avec un serveur http plus puissant, qui se chargera du contenu statique. Tomcat pourra ainsi être mis à contributioné uniquement pour les requêtes mettant en œuvre des servlets, et c'est pour ce principe que nous opterons [Chambon, 2002][[Web 6]][Oudomsouk, 2003].

1. 3. Serveurs de données

Concernant le serveur de bases données, le client émet des requêtes SQL sous forme de messages en direction du serveur. Le résultat de chaque requête SQL est renvoyé sur le réseau. Le code qui traite la requête ainsi que les données résident sur la même machine. Le serveur utilise sa propre capacité de traitement pour rechercher les données demandées au lieu de transmettre tous les articles au client et de laisser ce dernier les traiter. Dans notre cas nous avons opté pour le serveur MySQL.

Et ce car, MySQL est le plus populaire des serveurs de bases de données SQL Open Source

1. 4. Serveurs OLAP

On Line Analytical Processing : technologie d'analyse directe de données. Technologie de calcul et d'analyse de données à des fins commerciales ou de production, fondée sur des requêtes structurées suivant des critères combinés, appelés dimensions.

Le serveur utilisé dans notre application est *Mondrian.*

Mondrian est un serveur OLAP (On Line Analytical Processing) disponible sous licence open source.

Il fait partie de la catégorie des serveurs R-OLAP, c'est-à-dire qu'il accède à des données contenues dans une base relationnelle.

Mondrian exécute des requêtes utilisant le langage MDX, également utilisé dans Microsoft SQL Server. Ce langage permet de créer des requêtes dont l'équivalent en langue SQL nécessiterait un grand nombre de requêtes et des temps d'exécution beaucoup plus longs. [Web 4]

Lorsque Mondrian interroge l'entrepôt de données, il stocke les résultats dans un cache, et ce cache influence l'exécution des requêtes. Lorsque Mondrian est interrogé pour la première fois, le temps d'exécution de la requête est aléatoire selon la taille du résultat

renvoyé. Cependant, lorsque la même requête est exécutée deux fois, la seconde exécution prend quelques millisecondes.

Le MDX : MDX (Multi Dimensional eXpression) est un langage de requêtes pour les bases de données multidimensionnelles, de la même manière que SQL est utilisé pour les requêtes sur les bases de données relationnelles. Dans son approche, MDX est proche du SQL sur son aspect *select* et *where* même si la similarité ne va pas plus loin. Le but des expressions multidimensionnelles MDX est de rendre aisé et intuitif l'accès aux données de différentes dimensions.

MDX est fait pour naviguer dans les bases multidimensionnelles et pour définir des requêtes sur tous les objets (dimensions, hiérarchies, niveaux, membres et cellules) afin d'obtenir une représentation sous forme de tableaux croisés.

Il en découle une approche très hiérarchisée. Tout d'abord un cube est composé de dimensions. Une dimension peut contenir une ou plusieurs hiérarchies. Par exemple, la dimension "Time" contient deux hiérarchies : "Year, Quarter, Month" et "Year, Week, Day".

Le côté client de notre application renferme des clients de micro-navigateurs WAP supportées par des plates formes clientes *mobile* (téléphones portables, PDA,…)

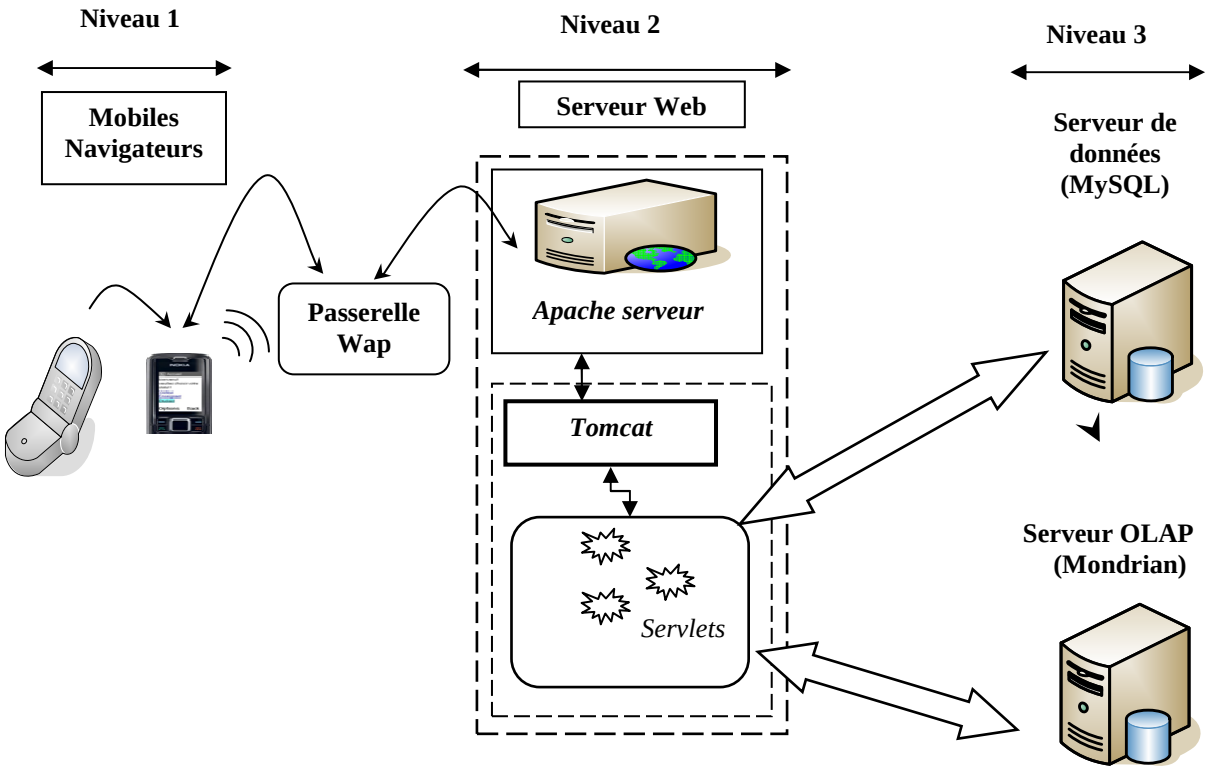

Fig. 6.2. – Architecture technique de l'application.

2. Environnement de développement

Le langage choisi pour le développement de l'application est le langage *JAVA*, et la technique utilisée est celle des *servlets*, contenues dans la plate-forme *Eclipse*. Les résultats de notre application seront dispensés sur un navigateur *WAP*, utilisant le langage *WML*.

2. 1. Qu'est ce que c'est qu'une servlet ?

Une servlet est une petite application Web Java dont le rôle est de recevoir des requêtes HTTP, de rechercher et de produire le résultat attendu puis de le retransmettre au demandeur par l'intermédiaire d'un serveur Web.

Pour créer, tester et par la suite utiliser les servlets, il faut disposer des éléments logiques suivants [Saumont, 2000] :

- Le client (Navigateur Web ou WAP quelconque),
- Le serveur/conteneur de servlets,
- Un système de développement (pour l'écriture des codes des programmes).

Les servlets n'ont pas d'interfaces graphiques. Elles sont exécutées sur des serveurs équipés d'un module spécial appelé conteneur de servlets, celui utilisé dans notre cas est Tomcat.

2. 2. Le WAP

Le WAP (pour Wireless Application Protocol) désigne l'ensemble des spécifications techniques issues du WAP Forum, une alliance créée en 1997 et regroupant les principaux acteurs du domaine des communications sans fil avec comme objectif de définir les protocoles de communications qui permettent de développer des applications et des services opérant sur des réseaux de communication sans fil. Le WAP a pour objectif d'ouvrir le monde du Web aux utilisateurs de téléphones portable ou PDA.

Le WAP, Wireless Application Protocol est une nouvelle technologie standard qui permet d'adapter les formats d'Internet aux contraintes des téléphones portables tels que :

- Débit ;
- Taille de l'écran, noir/blanc ;
- Possibilités limitées de saisie ;
- Vitesse de connexion relativement lente ;
- Peu de mémoire à disposition ;
- Logiciels simples.

Ce Protocol est utilisé dans notre réalisation, côté client, vu que ce dernier interroge notre système via une plate forme mobile répondant aux caractéristiques présentées ci-dessus.

2. 3. Présentation du langage WML

Le standard WAP et le langage l'accompagnant, WML, correspondent au couple HTTP/HTML.

Le langage WML a été conçu par les membres du WAP Forum4 pour décrire une page de manière simple et efficace. Le besoin d'un nouveau langage tenait au fait que le contexte et les contraintes d'utilisation des mobiles étaient très différents de ceux du Web.

Comme le HTML, le langage WML est aussi un langage de balises (markup) mais là s'arrête la similitude avec HTML car le WML est en fait dérivé du XML dont il reprend les règles et la syntaxe.

3. Mise en Œuvre de la solution proposée

3. 1. La base de données : Utilisateurs

La base de données *Utilisateurs* contient toutes les informations relatives à chaque utilisateur. Comme montré dans la figure 6.3, la base de données contient les tables suivantes:

- ***Users*** : cette table contient les utilisateurs de notre système regroupant les attributs suivants :
  - ***user_name*** : une chaine de caractères représentant le nom ou le pseudo nom de l'utilisateur ;
  - ***user_password*** : un mot de passe ;
  - ***taille_ligne*** : un nombre entier représentant le nombre maximal de cellules affichables sur la ligne ;
  - ***taille_colonne*** : est le nombre maximal de cellules que l'utilisateur veut afficher sur la colonne.
- ***dim_order*** : cette table contient les ordres sur les dimensions. Ces ordres font partie des préférences. Chaque ordre appartient à un utilisateur dont le nom est précisé par le champ *user_name*.
  - Le champ ***dim*** : contient le nom de la dimension et ;
  - ***Poids*** : est un nombre réel représentant le poids pondéré à la dimension *dim* selon les préférences de l'utilisateur user_name. Ce nombre est inversement proportionnel à l'ordre de préférence de cette dimension dim. C'est-à-dire soient par exemple deux dimensions dim_1, dim_2 et n_1, n_2 respectivement leurs poids. Si dim_1 est plus préférée à dim_2 alors $n_1 < n_2$

mem_order : cette table contient les ordres sur les membres. Chaque ordre est relié à une dimension qui est spécifiée par le champ ***dim***.

A chaque membre ***mem*** est associé un poids qui reflète l'ordre de préférence de ce membre. Le poids est inversement proportionnel à l'ordre de préférence de membre *mem*.

A noter que, nous avons utilisé une approche quantitative pour l'expression des préférences. Ceci est justifié par :

A notre avis, une approche qualitative :

- Impose une structure de tables (membre1, membre2,…) qui donne des tuples longs, et les valeurs (des membres ou de dimensions) seront positionnées dans les tables suivant l'ordre de préférences de l'utilisateur.
- Et une mise à jour des préférences engendrera des opérations très complexes.

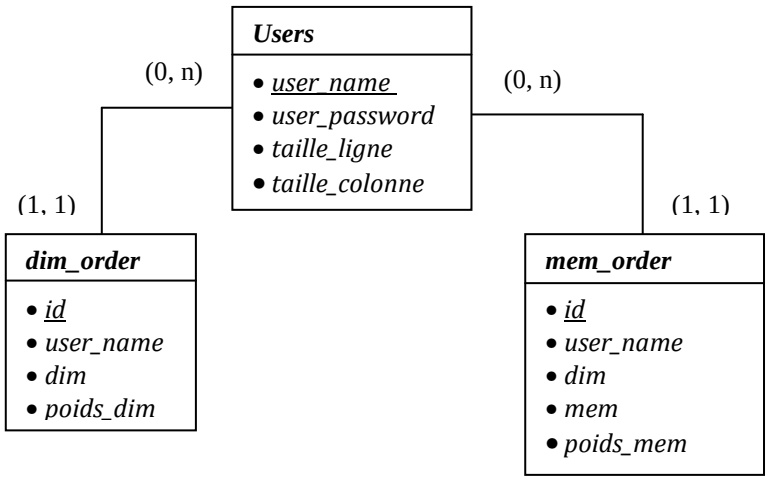

Fig. 6.3. – Diagramme de la base de données Utilisateurs.

3. 2. L'entrepôt de données

Les données à manipuler sont celles de « *FoodMart* ». Entrepôt de données de Microsoft. Cet entrepôt utilise un schéma en étoile illustré dans la figure 6.4. Il contient 7 cubes de données. Ci-dessous, les mesures et les dimensions de chaque cube. Les tests effectués sur notre système utilisent essentiellement le cube "*Sales*" car il contient la plus part des données. Et les préférences de l'utilisateur sont exprimées sur quelques dimensions de ce cube et leurs membres, enregistrées dans la base de données *users*, dans les tables *dim_order* et *mem_order*.

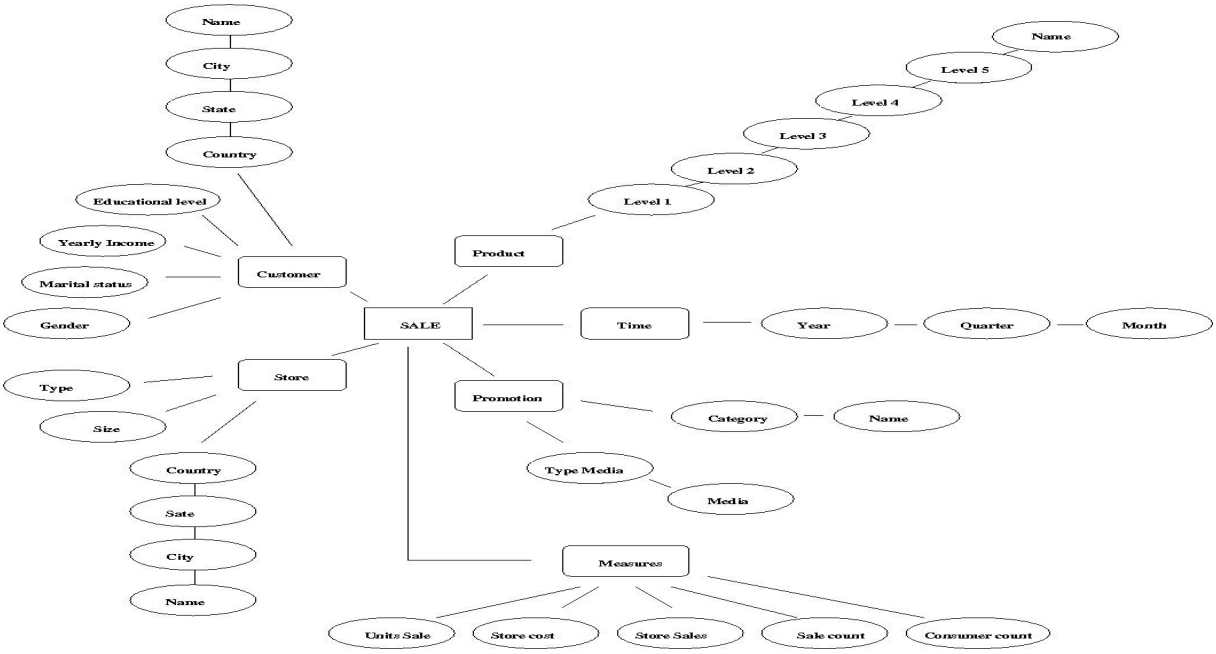

Fig. 6.4.– Diagramme en étoile de l'entrepôt de données Foodmart.

Cube	**Dimensions**
Warehouse	Store, Store Size in SQFT, Store Type, Time, Product, Warehouse
Store	Store, Store Type, Has coffee bar, Grocery sqft
HR	Time, Store, Pay Type, Store Type, Position, Department, Employees with Time being not the global one
Sales	Store, Store Size in SQFT, Store Type, Time, Product, Promotion Media, Promotions, Customers, Education Level, Gender, Marital Status, Yearly Income
Sales Ragged	Measures, Store, Geography, Store Size in SQFT, Store Type, Time , Product, Promotion Media, Promotions, Customers, Education Level, Gender, Marital Status, Yearly Income
Sales 2	Time , Product, Measures, Gender
Warehouse and Sales	Store, Warehouse, Store Type, Time, Product, Promotion Media, Promotions, Customers, Education Level, Gender, Marital Status, Yearly Income

Tab. 6.1 –Cubes et dimensions de l'entrepôt de données Foodmart.

Dans nos expérimentations, nous avons exploité le cube « Sales » et les préférences de l'utilisateur sont exprimées sur 4 dimensions, et pas sur la totalité de leurs membres.
La taille de chaque table en termes d'instances est décrite dans la Table 6.2.

Table	***Nombre d'enregistrement***
Customers	10281
Product	1560
Promotion	1864
Sales_fact_1997	86838
Time	730

Tab. 6.2 –Taille des tables utilisées.

3. 3. Présentation à l'utilisateur

L'application réalisée est à base d'un enchaînement d'écrans WML dont le principe de fonctionnement est expliqué ci-après.

1. A commencer par la page d'accueil qui renvoie la servlet d'identification de l'utilisateur (Fig. 6.5), puis si l'identification est incorrecte, un message d'erreur sera affiché, sinon

Fig. 6.5. – Ecran « login »

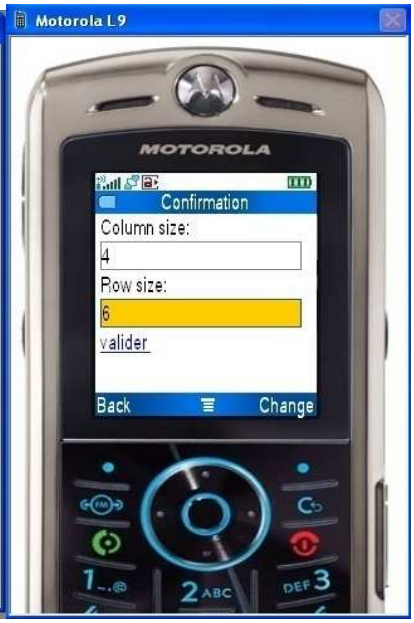

Fig. 6.6. – Ecran «taille écran »

2. Un écran récupère de la base de données user la taille de l'écran (nombre de lignes et de colonnes) de cet utilisateur, en lui donnant la main de modifier cette taille (Figure 6.6).

3. Plusieurs écrans sont présentés à l'utilisateur afin qu'il construit sa requête (Fig. 6.7),

Fig. 6.7. – Ecrans de construction de la requête

4. Avant que le serveur OLAP (Mondrian) exécute la requête construite sur l'entrepôt de données, cette requête est personnalisée selon les préférences de l'utilisateur extraites de la base de données *users*.

5. La servlet récupère le résultat de l'exécution de la requête, le traite et constitue la réponse qu'elle transmet au demandeur sous forme de tableau de cellules. Chaque cellule contient des coordonnées et un contenu, ce tableau sera affiché dans une page WML (Fig. 6.8).

Fig. 6.8. – Résultat de la requête personnalisée

4. Expérimentation

Le système de personnalisation reçoit une requête *MDX* provenant du client, ensuite, le processus de personnalisation doit personnaliser cette requête. Le principe de la personnalisation est de conserver le meilleur sous-ensemble de dimensions et de membres selon un profil donné.

De manière à valider notre approche sur la personnalisation de requêtes OLAP, des expérimentations ont été réalisées dans l'objectif de montrer l'impact positif de notre approche sur les techniques d'optimisation des requêtes. Et ainsi prouver la pertinence de notre méthode.

Pour évaluer l'efficacité de notre approche de personnalisation, nous comparons les temps d'exécution des requêtes personnalisées à ceux des requêtes non-personnalisées. Nous présentons, dans la section suivante, les résultats concernant les temps d'exécution de ces requêtes.

Requêtes avec et sans personnalisation

La mise en œuvre a pour objectif d'évaluer l'efficacité de notre approche de personnalisation de requêtes OLAP intégrant le profil de l'utilisateur dans le processus d'exécution de la requête. Cette évaluation consiste à comparer les temps d'exécution de requêtes personnalisées et requêtes non personnalisées, ainsi que le nombre de cellules généré dans les deux cas.

Les tests d'évaluation sont exécutés sur un ordinateur, sous Windows XP ayant un processeur Dual-Core de 1.6 GHz et 800 Mo de RAM.

Nous avons pris 5 requêtes qui renvoient 45, 165, 306, 6768 et 30843 cellules. Nous avons calculé leurs temps d'exécution. Ensuite, nous avons personnalisé ces requêtes selon les préférences des utilisateurs, puis nous avons exécuté les requêtes personnalisées. La figure 6.9 donne le gain en temps des requêtes personnalisées par rapport aux requêtes non personnalisées à la première exécution, c'est à dire sans l'utilisation de cache au sein du serveur *OLAP*.

Requêtes	***Nombre de cellules***	***Temps d'exécution (seconds)***	
		avant personnalisation	***après personnalisation***
select {[Measures].[Unit Sales], [Measures].[Store Cost], [Measures].[Store Sales]} ON COLUMNS, {([Promotion Media].members)} ON ROWS from [Sales]	45	5	4
select {[Measures].[Unit Sales], [Measures].[Store Cost], [Measures].[Store Sales]} ON COLUMNS, {([Product].[Product Category].members)} ON ROWS from [Sales]	165	7	5
select {[Measures].[Unit Sales], [Measures].[Store Cost], [Measures].[Store Sales]} ON COLUMNS, {([Product].[Product Subcategory].members)} ON ROWS from [Sales]	306	8	6
select {[Measures].[Unit Sales], [Measures].[Store Cost], [Measures].[Store Sales]} ON COLUMNS, {([Product].members)} ON ROWS from [Sales]	6768	14	10
select {[Measures].[Unit Sales], [Measures].[Store Cost], [Measures].[Store Sales]} ON COLUMNS, {([Customers].[Name].members)} ON ROWS from [Sales]	30843	19	12

Tab 6.3 –Temps d'exécution des requêtes et taille de la table résultat

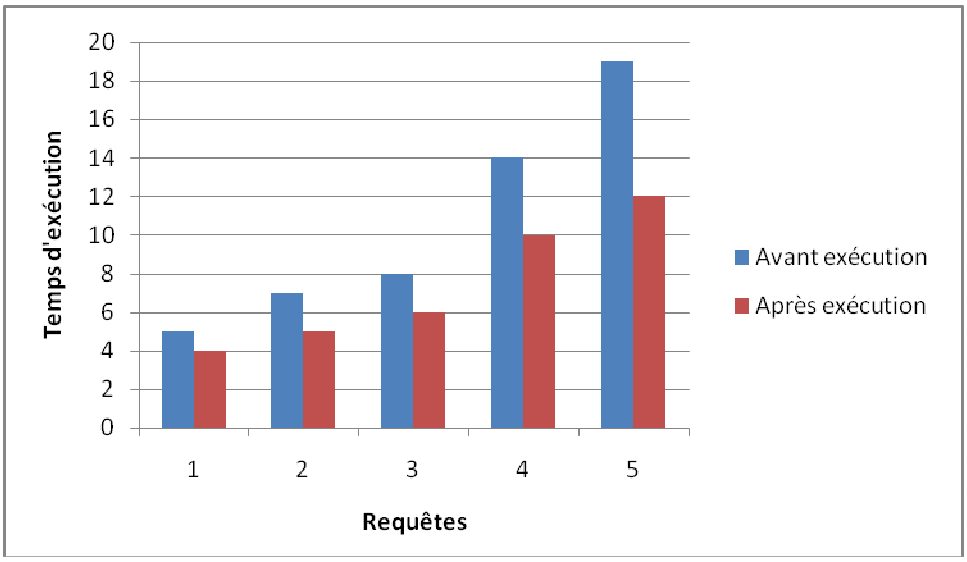

Fig 6.9. – Temps d'exécution

Nous remarquons qu'il existe bien un gain significatif en temps pour les requêtes personnalisées, cela est plus voyant quand la requête de départ contient un nombre de cellules important. Les temps d'exécution des requêtes personnalisées restent inférieurs aux temps d'exécution des requêtes non personnalisées, et cela pour n'importe quelle contrainte.

Cela prouve que notre algorithme de personnalisation agit positivement sur les techniques d'optimisation des requêtes utilisées par Mondrian.

5. Conclusion

Tout au long du présent chapitre, nous avons présenté et discuté le système développé ainsi que son fonctionnement et ce pour évaluer l'approche proposée.

Nous avons expérimenté notre approche et les tests que nous avons effectués ont révélé que notre algorithme de personnalisation donnait des résultats très satisfaisants, en termes de temps d'exécution et nombre de cellules généré par la requête.

Nos tests ont donc permis de nous rendre compte que l'algorithme de personnalisation que nous avons établi permet de générer des tuples préférés et visualisables en un temps d'exécution très réduit par rapport à l'exécution d'une requête MDX classique.

Les résultats expérimentaux sont encourageants et montrent la faisabilité de notre approche. Néanmoins, dans le but d'avoir de meilleures performances, des améliorations peuvent être apportées, et c'est ce que le chapitre suivant évoque.

7 Conclusion et perspectives

Au terme de ce rapport, nous présentons une synthèse sur les acquis dans le domaine de la personnalisation des requêtes et plus précisément dans le contexte des entrepôts de données. Et dessinons quelques perspectives ouvertes.

1. Conclusion

La personnalisation dans les bases de données multidimensionnelle, où L'entrepôt est conçu pour supporter des requêtes complexes de décision (requêtes OLAP) dont les résultats sont visualisés sous forme de tableaux croisés (ces résultats peuvent être très volumineux et souvent ils ne peuvent être visualisés entièrement sur le dispositif d'affichage (PDA, téléphone mobile, etc.)) se concentre sur la notion de visualisation.

Dans ce rapport, nous avons étudié dans un premier temps le profil utilisateur, car ce dernier constitue le noyau de la personnalisation, puis nous avons étudié la personnalisation dans les bases de données relationnelles, qui est le domaine le plus proche à celui des entrepôts de données. Ensuite nous avons présenté brièvement les différents travaux de personnalisation sur les entrepôts de données. Et comme l'objectif de notre travail est la « personnalisation de visualisation », nous nous somme concentré sur le travail de [Bellatreche et al, 2006], le seul travail recensé dans notre étude bibliographique qui traite cet aspect de personnalisation, la personnalisation de visualisation, méthode permettant de personnaliser les visualisations obtenues en réponse à des requêtes OLAP, sur la base de profil utilisateur. Cette méthode repose sur l'ordonnancement de références de cube générées par la requête utilisateur. Cet ordre est basé sur les préférences utilisateur contenues dans son profil.

Nous avons dégagé dans notre étude les points forts et les points faibles de cette méthode.

Et pour palier aux insuffisances décelées, nous avons proposé notre approche de personnalisation de visualisation se basant sur l'ordonnancement des membres et dimensions, et non pas sur l'ordonnancement des références.

Notre système de personnalisation reçoit une requête *MDX* provenant de l'utilisateur. Les étapes de la personnalisation consistent à :

- Extraire les dimensions de la requête *MDX* de chaque axe (ligne et colonne) ;
- Ordonner les dimensions de chaque axe selon les préférences de l'utilisateur contenues dans son profil;
- Affecter à chaque dimension d_i un nombre de membres affichables m_i et ce en :
    - Décomposant le nombre maximal de cellules affichable sur l'écran en facteurs.
- Extraire pour chaque dimension de chaque axe, les membres de la requête ;
- Ordonner ces membres selon les préférences utilisateur ;
- Prendre les m_i meilleurs membres de chaque dimension d_i ;
- Réécrire la requête initiale en prenant pour chaque dimension, ses meilleurs membres affichables.
- Exécuter la requête personnalisée, dont évidemment la réponse sera complètement visualisable sur l'écran de l'utilisateur.

Perspectives de recherche

Notre synthèse sur la personnalisation des requêtes dans le contexte des bases de données fait ressortir immédiatement les principales lignes de recherche.

Commençons par le modèle de préférence qui représente le noyau de la personnalisation qui consiste à étendre la modélisation de profil utilisateur contenant des préférences sur les visualisations à un profil tenant compte du coût d'exécution : temps et espace de stockage.

Malheureusement, l'intégration du profil dans le sens de rajouter des conditions de sélection à partir des préférences de l'utilisateur aux requêtes, pose des problèmes au niveau des techniques d'optimisation qui représente des notions très importantes dans un tel domaine. En effet, La suite naturelle à ce travail sur la personnalisation pourra être l'étude de la combinaison de la personnalisation avec les autres techniques d'optimisation de requêtes OLAP.

Bibliographie

[Abiteboul et al., 1995] Abiteboul, S., Hull, R., and Vianu, V. Foundations of Databases.Addison-Wesley. (1995).

[Agrawal and Wimmers, 2000] Agrawal, R. and Wimmers, E. L. A framework for expressing and combining preferences. In Chen, W., Naughton, J. F., and Bernstein, P. A., editors, Proceedings of the 2000 ACMSIGMOD International Conference on Management of Data, May 16-18, 2000, Dallas, Texas, USA, pages 297–306. ACM. (2000).

[Agrawal et al., 2003] Agrawal, S., Chaudhuri, S., Das, G., and Gionis, Automated ranking of database query results. In CIDR. . (2003).

[Axis, 2005] Axis (2005). Web services - axis. Available at http://ws.apache.org/axis/.

[Bellatreche et al, 2005] Bellatreche, L., Giacometti, A., Marcel, P., Mouloudi, H., and Laurent, D. Apersonalization framework for olap queries. In Song, I.-Y. and Trujillo, J., editors, DOLAP, pages 9–18. ACM. (2005).

[Bellatreche et al, 2006] Bellatreche, L., Giacometti, A., Marcel, P., and Mouloudi, H Personalization of mdx queries. In BDA’06 : 22èmes journées Bases de Données Avancées, Lille, 17-20 octobre 2006, Actes.

[Borzsonyi et al, 2001] Borzsonyi, S., Kossmann, D., and K.Stocker. The skyline operator. In IEEE : Proceedings of the IEEE International Conference on Data Engineering., pages 421–430. (2001)

[Bouzeghoub et al, 2005] Mokrane Bouzeghoub, Dimitre Kostadinov. Personnalisation de l'information : Aperçu de l'état de l'art et définition d'un modèle flexible de définition de profils. In Actes de la seconde édition de la Conférence en Recherche d'Information et Applications (CORIA), pages 201.218, Grenoble. (France, 2005).

[Bouzeghoub et al, 2004] Mokrane Bouzeghoub,Dimitre Kostadinov. Une approche multidimensionnelle pour la personnalisation de l’information. « INRIA Roocquencourt et laboratoire PRiSM, université de versailles 45, avenue des Etats-Unis,78035 Versailles ». (2004)

[Chambon, 2002] J-F Chambon, Réseaux sans-fil, Rapport de recherche, p.4-10, Ecole Nationale Supérieure des Mines - Saint Etienne – France. (Fev 2002).

[Chaudhuri and Dayal, 1997] Chaudhuri, S. and Dayal, U. An overview of data warehousing and olap technology. SIGMOD Record, 26(1) :65–74. (1997).

[Chaudhuri et al., 2004] Chaudhuri, S., Das, G., Hristidis, V., andWeikum, G. Probabilistic ranking of database query results. In VLDB'04 : Proceedings of the Thirtieth International Conference on Very Large Data Bases, Toronto, Canada, August 31 - September 3 2004, pages 888–899. (2004).

[Chomicki, 2002] Chomicki, J. Querying with intrinsic preferences. In Proceeding of the 8th InternationalConference on Extending Database Technology, Prague, Czech Republic, pages 34–51. (2002).

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

[Colonna, 2002] F.M Colonna, L'architecture client/serveur. http ://www.lsis.org/˜colonnafm/docs/SauvegardeTPS/TPS Dea/Architectures-Client-Serveur.pdf. (octobre 2002)

[Favre, 2007] Cecile Favre. Évolution de schémas dans les entrepôts de données : mise à jour de hiérarchies de dimension pour la personnalisation des analyses. Thèse de doctorat, Université Lumière Lyon 2 École Doctorale Informatique et Information pour la Société (2007)

[Gallo, 2002] F. DI Gallo, Intégration des systèmes client/serveur, Cours du cycle d'approfondissement, CNAM Aix-en-Prrovence, France. (2001/2002).

[Giacometti et al, 2008] Arnaud Giacometti, Patrick Marcel, Elsa Negre A Framework for Recommending OLAP Queries**.** In 11th ACM International Workshop on Data Warehousing and OLAP (DOLAP 08), Napa Valley, California, USA, pp. 73–80. Laboratoire d'Informatique Université François Rabelais de Tours – France (2008)

[Golfarelli and Rizzi, 1998] Golfarelli, M. and Rizzi, S. (1998). Methodological framework for datawarehouse design. In DOLAP'98, ACMFirst InternationalWorkshop on DataWarehousing and OLAP, November 7, 1998, Bethesda, Maryland, USA, Proceedings, pages 3–9. ACM.

[House, 2004] Well Consultants Samples. Tomcat Overview. http ://www.wellho.net/downloads/A651.pdf. (2004)

[Inmon, 1996] W H Inmon. Building the Data Warehouse. Third ed..John Wiley & Sons. (1996).

[Jagadish et al., 1999] Jagadish,H.V., Lakshmanan, L.V. S., and Srivastava,D. What can hierarchies do for data warehouses ? In VLDB'99, Proceedings of 25th International Conference on Very Large Data Bases, September 7-10, 1999, Edinburgh, Scotland, UK, pages 530–541. . (1999)

[Jerbi et al,.2008] Jerbi, H., F. Ravat, O. Teste, et G. Zurfluh Management of Context-Aware Preferences in Multidimensional Databases. In ICDIM 08, pp. 669-675. (2008).

[Jerbi et al,.2009] Jerbi, H., F. Ravat, O. Teste, et G. Zurfluh. Applying Recommendation Technology in OLAP Systems. In ICEIS 09, pp. 220-233. (2009).

[Kießling, 2002] Kießling,W. Foundations of preferences in database systems. InVLDB : Proceedings of 28th International Conference on Very Large Data Bases, pages 311–322. (2002).

[Kießling, 2005] Kießling,W.. Preference queries with sv-semantics. In COMAD, pages 15–26. (2005)

[Kimball, 1996] Kimball, R. The Data Warehouse Toolkit. John Wiley & Sons. (1996).

[Kobs, 2001] Kobsa A. «Generic User Modeling Systems », Journal On User Modeling and User-Adapted Interaction, vol. 11, 2001, P49-63. (2001)

[Kossmann, 2001] S. Borzsonyi, D. Kossmann, K. Stocker, The Skyline Operator, IEEE Conf. On data Engineering. (2001).

[Koutrika and Ioannidis, 2004] Koutrika, G. and Ioannidis, Y. E. Personalization of queries in database systems. In ICDE '04 : Proceedings of the 20th International Conference on Data Engineering, pages 597–608, Washington, DC, USA. IEEE Computer Society. (2004).

[Koutrika et Ioannidis 2005a] Georgia Koutrika, Yannis Ioannidis; Personalization Queries under a Generalized preference Model Athens Univ., Greece; Data Engineering, 2005. ICDE 2005. Proceedings. 21st International Conference on Date de publication : 2005 page: 841- 852 ISSN: 1084-4627
ISBN: 0-7695-2285-8. (5-8 Avril 2005)

[Koutrika et Ioannidis 2005b] Koutrika, G.; Ioannidis, Y. Constrained Optimalities in Query personalization*. Date de publication : (2005)
Pages: 73 – 84 ISBN:1-59593-060-4 ACM Press New York, NY, USA.

[Lacroix 1987] M. Lacroix, P. Lavency, Preference: Putting More Knowledge into Queries, Proceeding of the 13th VLDB Conference, Brighton. (1987).

[Lechani et al, 2005] Lynda Lechani Tamine, Mohand Boughanem. Accès personnalisé à l'information : vers la définition d'un profil utilisateur multidimensionnel. In International Symposium On Programming Systems, pages 20.28. USTHB. (2005)

[McTear, 1993] McTear M. F. User modelling for adaptative computer systems : A survey of recent developments. Artificial Intelligence Review, 7 : P.157-184. (1993).

[Missaoui et al, 2007] R.Missaoui, G Jatteau, A Boujenoui, and S Naouali. Towards Integrating Data Warehousing with Data Mining Techniques, pages 253–276. Data Warehouses and OLAP : Concepts, Architectures and Solutions. Idea Group Publishing. (2007).

[Mouloudi, 2007] Hassina MOULOUDI. Personnalisation de requêtes et visualisations OLAP sous contraintes. Thèse de doctorat. École Doctorale : Santé, Sciences et Technologies.
Année Universitaire : 2006-2007

[Oudomsouk, 2003] P. Oudomsouk, J. Marangone, J. Gaffier, P.M. Favennec, Plate-forme de push service, projet ERACE, Institut des applications avancées de l'internet. (2003).

[Ravat et Teste 2008] Ravat, F. et O. Teste Personalization and OLAP Databases. New Trends in Data Warehousing and Data Analysis, 3, 71–92. (2008).

[Romero and Abelló, 2007] Romero, O. and Abelló, A. On the need of a reference algebra for olap. In Song, I. Y., Eder, J., and Nguyen, T. M., editors, DaWaK, volume 4654 of Lecture Notes in Computer Science, pages 99–110. Springer. (2007).

[Saumont, 2000] P.Y. Saumont, A. Mirecourt, Servlets et Java Server Page Le guide du développeur, OSMAN EYROLLES MULTIMEDIA (2000).

[Tamine et al, 2007] Lynda Tamine-Lechani, Nesrine Zemirli, Wahiba Bahsoun. Approche statistique pour la définition du profil d'un utilisateur de système de recherche d'information. Information – Interaction - Intelligence, 7(1), Laboratoire PRiSM, Université de Versailles (2007)

[Tamine et al, 2008] Lynda Tamine, Wahiba Bahsoun. Définition d'un profil multidimensionnel de l'utilisateur : Vers une technique basée sur l'interaction entre dimensions. Programme ACI Masses de Données, projet MD-33 (http ://apmd.prism.uvsq.fr/). Laboratoire IRIT (Equipe SIG) Université Paul Sabatier (2008)

[Zemirli, 2008] WAHIBA NESRINE ZEMIRLI Modèle d'accès personnalisé à l'information basé sur les diagrammes d'Influence intégrant un profil utilisateur évolutif. Thèse de doctorat, Université Paul Sabatier de Toulouse III (2008)

Références Webographiques

[Web 1]: http://maia.loria.fr/lexique/profil_utilisateur.html.

[Web 2]: http://www.serveur-dedie.com/serveur/serveurs/15-serveur-d-application.html

[Web 3]: http://www.journaldunet.com/encyclopedie/definition/972/34/20/tomcat.shtml

[Web 4]: http://decisionnel-open-source.smile.fr/les-composants-decisionnels/mondrian

[Web 5]: http ://www.centralweb.fr

[Web 6]: http ://www.minefi.gouv.fr