Personnalisation de l'information : une approche de gestion de profils et de reformulation de requêtes

Dimitre Kostadinov

THESE

présentée à
L'UNIVERSITE DE VERSAILLES
SAINT-QUENTIN-EN-YVELINES

pour obtenir le titre de

DOCTEUR EN INFORMATIQUE

soutenue par
Dimitre Kostadinov

Titre

Personnalisation de l'information : une approche de gestion de profils et de reformulation de requêtes

Data Personalization: an approach for profile management and query reformulation

Jury

Jean-Marc Petit	Professeur des Universités, INSA	Rapporteur
Daniel Rocacher	Maître de Conférence, HDR, ENSSAT	Rapporteur
Mohand Boughanem	Professeur des Universités, Toulouse	Examinateur
Michel Scholl	Professeur des Universités, CNAM Paris	Examinateur
Stéphane Lopes	Maître de Conférence, UVSQ	Examinateur
Mokrane Bouzeghoub	Professeur des Universités, UVSQ	Directeur de thèse

Remerciements

Je souhaite remercier très particulièrement mon directeur de thèse Monsieur Mokrane Bouzeghoub. Je le remercie de m'avoir fait confiance et de m'avoir fourni les meilleures conditions de travail possibles. Les discussions et le temps qu'il m'a accordés tout au long de ma thèse m'ont permis d'avancer dans mon travail et ont contribué à améliorer mes connaissances. Ses remarques ont toujours été claires et précises ce qui a facilité le processus d'apprentissage du métier. Ses conseils aussi bien sur le plan professionnel que personnel m'ont toujours été d'une grande aide. Pour tout ce que vous avez fait pour moi, merci du fond du cœur !

Je voudrais également remercier Stéphane Lopes avec qui nous avons beaucoup collaboré durant ma thèse. Les discussions que nous avons eues m'ont permis d'avancer plus rapidement dans mon travail.

Je remercie mes rapporteurs Jean-Marc Petit et Daniel Rocacher pour la rapidité avec laquelle ils ont lu et évalué mon manuscrit ainsi que pour l'intérêt qu'ils ont porté à mon travail. Leurs remarques ont été très constructives et m'ont aidé à améliorer mon mémoire. Je souhaite également remercier les autres membres du jury de ma thèse Michel Scholl et Mohand Boughanem d'avoir accepté de juger ce travail.

Je tiens à remercier tous les autres membres de l'équipe dans laquelle j'ai travaillé durant ma thèse Zoubida Kedad et Daniela Grigori ainsi que tous mes collègues qu'ils soient actuels ou ex : Xiaohui Xue, Veronika Peralta, Assia Kadi, Juan-Carlos Corales et Sofiane Abbar. Ils m'ont reçu très chaleureusement et m'ont permis de travailler dans une ambiance très amicale et accueillante.

Je voudrais remercier l'administration de l'université de Versailles qui m'a guidé dans mes démarches administratives.

Je souhaite exprimer toute ma gratitude envers ma famille, mon père David Kostadinov, ma mère Zhana Kostadinova et ma sœur Irena Kostadinova ainsi que ma compagne Petia Nikolova pour m'avoir été d'un grand soutien moral durant toute ma thèse.

Je remercie également tous mes amis avec qui j'ai passé de très agréables moments de détente.

Résumé

Cette thèse contient deux parties. La première est orientée vers l'étude de l'état de l'art sur la personnalisation et la définition d'un modèle de profil utilisateur. La seconde partie se focalise sur la reformulation de requêtes guidée par le profil utilisateur.

La personnalisation a pour objectif de faciliter l'expression du besoin utilisateur et de rendre l'information sélectionnée intelligible à l'usager et exploitable. Elle 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. L'ensemble de ces informations est représenté dans un modèle d'utilisateur appelé souvent profil. Le premier travail de cette thèse est de proposer un modèle ouvert de profil capable d'acquérir toutes les informations décrivant l'utilisateur.

La personnalisation de l'information intervient à toutes les étapes du cycle de vie d'une requête. La seconde contribution de cette thèse est l'étude de deux approches de reformulation de requêtes basées sur des techniques de réécriture et d'enrichissement existants et la proposition d'une approche de reformulation avancée qui alterne des étapes d'enrichissement et de réécriture. Les trois approches sont évaluées sur un benchmark défini dans la thèse.

Abstract

This thesis contains two parts. The first one is a study of the state of the art on data personalization and a proposition of a user profile model. The second one is a focus on a specific problem which is the query reformulation using profile knowledge.

The goal of personalization is to facilitate the expression of the need for a particular user and to enable him to obtain relevant information when he accesses an information system. The relevance of the information is defined by a set of criteria and preferences specific to each user or community of users. These criteria describe the user's domain of interest, the quality level of the data he is looking for or the modalities of the presentation of this data. The data describing the users is often gathered in the form of profiles. In this thesis we propose a generic and extensible model of profile, which enables the classification of the profile's contents.

Personalization may occur in each step of the query life cycle. The second contribution of this thesis is the study of two query reformulation approaches based on algorithms for query enrichment and query rewriting and the proposition of an advanced query reformulation approach. The three reformulation approaches are evaluated on a benchmark described in the thesis.

Table des matières

Annexe 5: State of the Art

Liste des figures

Figures de l'Annexe 5

Liste des tableaux

Liste des Annexes

Chapitre 1. Introduction

1. Problématique et motivations

Les systèmes d'information actuels donnent accès à un grand nombre de sources hétérogènes et distribuées. Au fur et à mesure que les sources se multiplient et que le volume de données disponibles s'accroît, l'utilisateur se voit confronté à une surcharge informationnelle dans laquelle il est difficile de distinguer l'information pertinente de l'informations secondaire et même du bruit. En outre, l'évaluation d'une requête se fait généralement sans tenir compte du contexte et/ou des besoins spécifiques de l'utilisateur qui l'a émise. La même requête, faite par deux utilisateurs différents, produit les mêmes résultats même si ces utilisateurs n'ont pas les mêmes attentes.

Pour répondre aux problèmes de la surcharge informationnelle et pour pouvoir discriminer les utilisateurs en fonction de leurs besoins spécifiques, certains systèmes proposent des techniques de personnalisation basées sur le profil de l'utilisateur. La notion de profil modélise les centres d'intérêt de l'utilisateur ainsi que ses principales préférences en termes de filtrage et de qualité de l'information, de modalités d'accès aux systèmes et de livraison des résultats, du contexte géospatial dans le lequel il se trouve ainsi que des médias d'interaction qu'il utilise. L'ensemble de ces connaissances est défini en totalité ou en partie directement par l'utilisateur ou dérivé par des algorithmes d'apprentissage à partir des actions passées de l'utilisateur.

La classification, l'organisation et la structuration des données de profils est un élément clé de la personnalisation. Différents travaux ont abordé cet aspect sans le couvrir dans son ensemble. Par exemple, P3P [CDE+05], standard pour la sécurisation des profils, permet de définir des classes distinguant entre les *attributs démographiques*, *les attributs professionnels* et *les attributs de comportement*. Dans [AS99], les auteurs proposent un modèle de profil pour les utilisateurs d'une bibliothèque digitale, composé de cinq catégories d'informations : *Données Personnelles*, *Données Collectées*, *Données de Livraison*, *Données de Comportement* et *Données de Sécurité*. Ces tentatives de structuration sont louables mais insuffisantes pour couvrir le champ de la personnalisation. Par ailleurs, elles se contentent de catégoriser les informations de profil, mais sont difficilement extensibles.

La personnalisation de l'information constitue un enjeu majeur pour l'industrie informatique. Que ce soit dans le contexte des systèmes d'information d'entreprise, du commerce électronique, de l'accès au savoir et aux connaissances ou même des loisirs, la pertinence de l'information délivrée, son intelligibilité et son adaptation aux usages et préférences des clients constituent des facteurs clés du succès ou du rejet de ces systèmes.

La personnalisation de l'information a été particulièrement abordée dans trois domaines technologiques : l'Interaction Homme-Machine (IHM), la Recherche d'Information (RI), et les Bases de Données (BD).

Dans le domaine des IHM [EP00, LS01], la personnalisation se focalise principalement sur le niveau d'expertise et le métier de l'utilisateur afin de déterminer le type de dialogue que le système va avoir avec lui, les métaphores graphiques les plus appropriées ainsi que les modalités de livraison des résultats qu'il attend du système d'information.

Dans le domaine de la RI [CN04, BRS00, SL01, FS01, ZTB07], l'utilisateur fait partie du processus de personnalisation. L'évaluation d'une requête se fait généralement de façon interactive et incrémentale; à chaque itération, le système tient compte des informations collectées à partir des interactions précédentes avec l'utilisateur ou profite de l'expérience des autres utilisateurs (filtrage collaboratif). La personnalisation est ainsi définie comme un apprentissage réalisé à partir des préférences rendues par les utilisateurs à l'issue de la présentation des résultats successifs du système.

Dans le domaine des BD [Kie02 ; Kie05, KI04a, KI05b, Cho02, BKS01, LL87, RL06], l'utilisateur ne fait pas partie du processus de recherche d'informations. La requête contient en général l'ensemble des critères considérés nécessaires à produire des données pertinentes. Les profils sont alors intégrés directement aux requêtes par les utilisateurs ou lors de la compilation de ces dernières; ils sont alors pris en compte en une seule fois durant l'exécution de la requête. L'approche proposée dans cette thèse est faite dans ce domaine.

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 [BHK98, CT02, HCO+02, Paz99, PML05, MDS01, NDB06] 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. Ce mode opératoire est aussi appelé mode *push*. Le feedback des utilisateurs est très important pour affiner leurs profils et augmenter l'efficacité du système.

La personnalisation en interrogation [BRS00, KI04a, KI05b, Cho02, Kie02, Kie05, BKS01, ...] 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 enrichissant sa requête afin de la rendre plus précise [KI04a, KI05b], en choisissant les sources de données en fonction des exigences de qualité de l'utilisateur [NFS98] ou en personnalisant l'affichage des résultats [PG99]. Ce mode opératoire est également appelé mode *pull*. Le travail décrit dans cette thèse s'inscrit dans cette approche.

2. Contexte de la thèse

Le travail présenté dans cette thèse s'inscrit dans le contexte du projet Accès Personnalisé à des Masses de Données (APMD)[1]. Les objectifs du projet sont organisés en quatre sous-projets:

1. la modélisation et l'évolution des profils,
2. l'exécution adaptative de requêtes,
3. l'influence de la qualité sur la personnalisation,
4. l'évaluation et la validation des approches proposées dans le projet.

Le premier sous-projet nécessite une étude aussi large que possible de l'état de l'art sur la personnalisation. Le résultat attendu de cette étude est l'élaboration d'une typologie des connaissances constituant le profil, l'étude des formalismes de définition et de représentation des profils et la proposition de techniques de construction et d'évolution des profils.

[1] ACI Masses de Données : Projet MD-33 Accès Personnalise à des Masses de Données. http://apmd.prism.uvsq.fr/

Le second sous-projet concerne l'exploitation du contenu du profil utilisateur dans le cycle de vie d'une requête. Le cycle de vie d'une requête est composé de quatre phases : la reformulation, l'optimisation, l'exécution et la présentation des résultats à l'utilisateur. L'objectif de ce sous-projet est de proposer des algorithmes qui permettent de tenir compte des informations du profil utilisateur pendant chacune de ces phases. Ces algorithmes doivent être définis dans un contexte de système d'informations distribué et hétérogène, donnant accès à des sources d'informations pouvant être aussi bien des bases de données relationnelles que des bases documentaires.

Le troisième sous-projet est lié à l'étude des relations qui existent entre la qualité et la personnalisation. Il comprend trois objectifs : (i) la définition de facteurs de qualité qui ont un impact sur la personnalisation, (ii) la prise en compte de ces facteurs dans le cycle de vie de la requête et (iii) la validation d'un profil qui consiste à vérifier si les préférences et les exigences d'un utilisateur peuvent être satisfaites dans un environnement technique et à quel point.

Finalement, le quatrième sous-projet a pour objectif de valider par des expérimentations les approches proposées dans les trois autres sous-projets. Ceci doit être fait en implémentant les algorithmes et en élaborant des scénarios d'évaluations. Le projet APMD vise deux scénarios d'évaluation : un premier qui doit montrer l'impact du profil sur la réécriture de requêtes et un second qui doit mesurer l'influence du profil sur l'exécution sémantique des requêtes.

L'architecture de référence du projet est illustrée par la Figure 1.1.

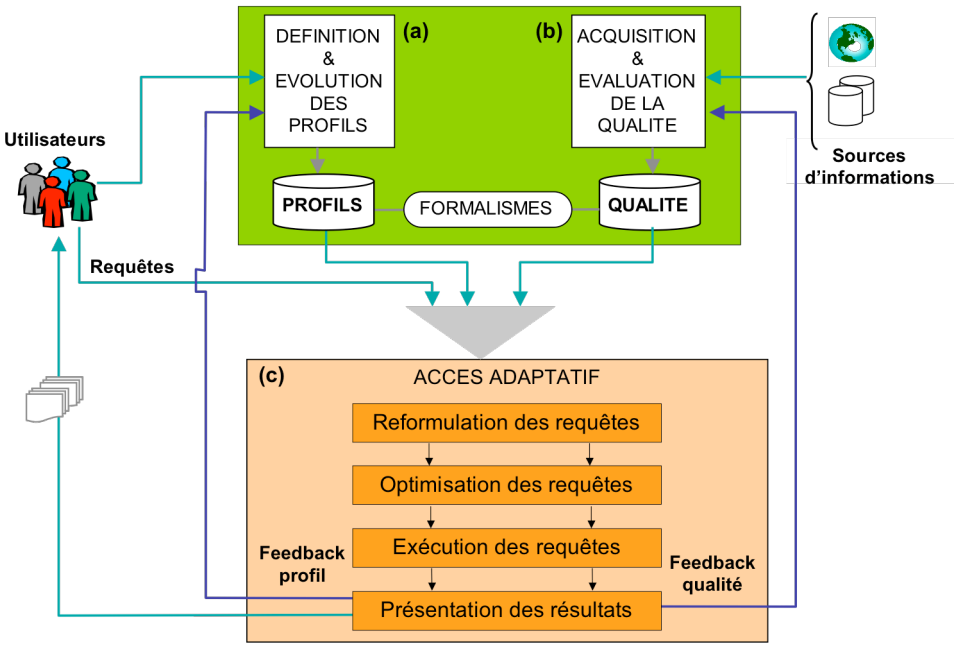


Figure 1.1 : Composants fonctionnels de APMD

L'objectif du travail décrit dans cette thèse est double :

- Analyser l'état de l'art et établir un modèle de référence pour la définition des profils. Ce modèle doit constituer le support à une plateforme de gestion de profils, offrant des opérations d'import/export de profils, de spécialisation/instanciation de profils, d'appariement et d'évolution de profils.
- Proposer des algorithmes de reformulation de requêtes guidés par le profil utilisateur. En particulier, montrer comment, dans un cadre distribué, le profil

impacte le choix des sources de données, la réécriture des requêtes initiales et la qualité des résultats produits.

3. Objectifs de la thèse

Les objectifs de cette thèse sont de deux ordres : la modélisation et la gestion des connaissances décrivant l'utilisateur d'une part, et l'exploitation de profils utilisateurs dans la reformulation et l'exécution de requêtes d'autre part.

Modélisation et gestion des connaissances décrivant l'utilisateur et son contexte

La première partie de la thèse est guidée par des objectifs liés à une contribution générale à la fois en état de l'art et en modélisation et gestion des profils et des contextes.

Le premier objectif de cette partie de la thèse est de proposer une étude assez large de l'état de l'art sur la personnalisation. Nous voulons montrer la diversité des approches de personnalisation à trois niveaux : (i) la définition et la représentation des profils, (ii) les techniques de construction de profils et (iii) l'exploitation les profils. Cette étude vise à illustrer le manque d'un cadre global de définition et de gestion de profils et de montrer qu'il y a autant de modèles de profils que d'applications.

Le second objectif est de proposer une modélisation des connaissances utilisées par les systèmes afin de fournir des services personnalisés. Nous voulons distinguer les notions de *profil*, de *contexte* et de *préférence* pour les étudier séparément et montrer clairement les relations sémantiques qui existent entre elles. Cette étude vise à nous mener à l'établissement d'un cadre de référence pour définir et manipuler des systèmes d'accès personnalisés à des masses de données. L'objectif final est de présenter ce cadre de référence sous forme d'un méta modèle extensible et adaptable à chaque type d'application.

Le troisième objectif de cette partie de la thèse est de proposer un ensemble d'opérations de gestion de profils et de contextes. Parmi les opérations possibles, nous nous focaliserons sur *l'instanciation* et sur *l'appariement* de profils en raison de leur caractère crucial pour la définition des autres opérations. Nous voulons étudier deux niveaux d'instanciation d'un profil (ou d'un contexte) : (i) instanciation du méta modèle pour obtenir un modèle type de profils et (ii) instanciation d'un modèle type pour obtenir une instance de profil particulière. L'objectif principal de l'appariement des profil est la proposition de primitives qui permettent de comparer deux profils pour vérifier leur équivalence, pour trouver leurs parties communes ou pour énumérer leurs différences. Ces primitives doivent s'appliquer aussi bien sur la structure des profils qu'aux valeurs de leurs attributs.

Exploitation de profils utilisateur dans la reformulation des requêtes

La seconde partie de la thèse est guidée par l'objectif d'apporter une solution à un problème particulier qui est la prise en compte du profil utilisateur dans la reformulation de requêtes. Le travail réalisé dans cette partie doit être situé dans un contexte distribué où le système d'information donne accès à un grand nombre de sources par réécriture de requêtes exprimées sur un schéma global et où la personnalisation se fait par enrichissement des requêtes avec des prédicats contenus dans le profil utilisateur. Cette partie de la thèse se divise en trois objectifs principaux : (i) la proposition d'approches de reformulation de requêtes basées sur des algorithmes d'enrichissement et de réécriture existants, (ii) la définition d'une nouvelle approche de reformulation de requêtes et (iii) l'évaluation des approches proposées.

Le premier point concerne la définition et l'étude de deux approches séquentielles de reformulation de requêtes. Elles peuvent être obtenues par l'exécution séquentielle d'algorithmes d'enrichissement et de réécriture. Il est donc nécessaire de choisir un algorithme de réécriture et un algorithme d'enrichissement. Pour pouvoir montrer les

avantages et les inconvénients des approches nous allons définir des métriques d'évaluation. Ces métriques doivent permettre de mesurer la part du profil utilisateur pouvant être prise en compte *(couverture du profil)* et *l'utilité* des prédicats du profil que l'approche utilise dans la phase d'enrichissement de la requête.

Le second point de cette partie de la thèse vise à proposer un nouvel algorithme de reformulation de requêtes. L'objectif de cet algorithme est de gommer les inconvénients des approches séquentielles tout en préservant leurs avantages. Il doit permettre de mieux tenir compte du profil utilisateur et d'éliminer le plus vite possible les reformulations qui ne sont pas pertinentes pour l'utilisateur.

Finalement, le troisième point est l'évaluation des trois approches de reformulation proposées. Pour atteindre cet objectif, il faut définir un benchmark et proposer des métriques d'évaluation qui permettent de faire des mesures à deux niveaux : niveau sémantique (compilation des requêtes) et niveau exécution des requêtes.

4. Résumés des chapitres et contributions

Le Chapitre 2 présente l'état de l'art sur la personnalisation. Il donne des exemples d'applications de personnalisation et montre les principales caractéristiques des systèmes de personnalisation. Il propose également une étude des différentes représentations des profils dans la littérature et discute de la typologie des préférences. L'algorithmique liée à la construction et à l'exploitation des profils est également étudiée. Les techniques de construction des profils montrent comment dériver les informations du profil à partir de la trace (logs) des interactions entre les utilisateurs et le système. Les approches d'exploitation des profils auxquelles nous nous sommes intéressés sont l'introduction des préférences dans les langages de requêtes, l'enrichissement des requêtes, la prise en compte des préférences de l'utilisateur dans la réécriture de requêtes et le matching entre les profils et le contenu des données.

L'étude de l'état de l'art sur la personnalisation nous a permis d'identifier un certain nombre de problèmes de recherche : la modélisation des connaissances décrivant l'utilisateur, la réutilisation de profil utilisateur d'un système à un autre, la construction automatique des profils, l'évolution du contenu du profil, l'exploitation du contenu du profil utilisateur. Nous concluons cet état de l'art en situant notre travail par rapport à ces problèmes. Concrètement, nous nous focalisons sur la modélisation des connaissances décrivant l'utilisateur et l'exploitation du contenu du profil utilisateur.

Le Chapitre 3 décrit notre contribution sur la modélisation et la gestion des connaissances décrivant l'utilisateur [BK05].

Nous proposons une approche dans laquelle les notions de *profil*, de *contexte* et de *préférence* sont étudiées séparément. Chaque notion est représentée par un méta modèle générique et extensible capable d'acquérir l'ensemble des connaissances la composant. La corrélation entre les trois méta modèles est décrite par les relations sémantiques qui les relient. Pris ensemble, ils permettent d'obtenir un méta modèle qui décrit entièrement l'utilisateur et qui peut être adapté à chaque application.

La gestion des connaissances décrivant l'utilisateur est présentée par un ensemble d'opérations de manipulation de profils, de contextes et de préférences. Ces opérations permettent d'instancier un profil, un contexte ou une préférence et d'apparier deux profils. L'instanciation se fait en deux temps : premièrement le méta modèle est instancié ce qui permet d'obtenir un type de modèle et deuxièmement le type de modèle est utilisé pour obtenir une ou plusieurs instances particulières. L'appariement de profils est constitué de trois

primitives qui permettent de déterminer si deux profils sont équivalents, de trouver leurs points communs ou de souligner leurs différences. Ces primitives s'appliquent aussi bien sur la structure des profils que sur leurs valeurs.

Les méta modèles ainsi que les opérations de manipulation de profils ont été implémentés dans une plateforme de gestion des connaissances [KPS+04], [KPS+05].

Le Chapitre 4 définie et évalue deux approches de reformulation de requêtes [KBL07b].

Nous nous plaçons dans un contexte où le système doit tenir compte à la fois de la distribution des sources de données et des préférences de l'utilisateur. Les approches que nous proposons dans ce chapitre sont basées sur des algorithmes de réécriture et d'enrichissement de requêtes existants qui permettent d'adresser respectivement la distribution des données et la personnalisation. La différence entre les deux approches vient de l'ordre dans lequel les deux algorithmes sont appliqués. Nous obtenons ainsi une approche enrichissement-réécriture et une approche réécriture-enrichissement.

La comparaison entre les deux approches est faite sur la base de deux métriques qui sont la couverture du profil utilisateur et l'utilité des prédicats du profil utilisateur pris en compte par l'approche. L'évaluation montre que les deux approches sont complémentaires. L'approche enrichissement-réécriture permet d'obtenir une bonne couverture du profil utilisateur, mais présente un risque de prise en compte de prédicats inutiles. L'approche réécriture-enrichissement possède une excellente utilité, mais peut avoir une faible couverture.

Le Chapitre 5 décrit notre approche avancée de reformulation de requêtes qui permet d'éviter les inconvénients des deux approches présentées dans le chapitre 4 tout en conservant leurs avantages [LBL07a]. L'idée principale de cette approche est d'alterner des étapes d'enrichissement et de réécriture afin de mieux tenir compte du profil utilisateur. Elle est composée de quatre étapes : expansion de la requête, identification des sources pertinentes, combinaison des sources pertinentes et enrichissement final. Notre contribution se situe dans la première et la troisième étape.

L'expansion de la requête permet de rendre exploitable le profil utilisateur. Elle consiste à étendre la requête initiale avec des relations virtuelles sur lesquelles sont exprimés les prédicats du profil utilisateur. Dans notre approche, l'expansion de la requête est présentée comme un problème de graphe. La requête utilisateur est vue comme un sous-graphe du graphe représenté par le schéma virtuel. Chaque nœud de ces graphes est une relation virtuelle et chaque arête est une contrainte référentielle (jointure). L'expansion de la requête revient à étendre son graphe avec des nœuds qui représentent des relations virtuelles sur lesquelles sont exprimés les prédicats du profil utilisateur. Nous proposons un algorithme qui permet de maximiser la part exploitable du profil utilisateur tout en minimisant le nombre de relations ajoutées à la requête.

La combinaison de sources contributives permet d'obtenir des réécritures candidates. Nous proposons un algorithme qui permet d'intégrer le profil utilisateur dans cette phase de la reformulation afin de produire uniquement des reformulations qui sont pertinentes pour l'utilisateur. Cet algorithme utilise une stratégie d'élagage qui permet d'éliminer en une seule fois plusieurs combinaisons de sources contenant le même sous-ensemble non pertinent de sources.

Le Chapitre 6 résume les évaluations que nous avons faites sur les trois approches de reformulation de requêtes présentées dans les chapitre précédents. Ce chapitre contient d'une

part la description du benchmark qui a servi pour les tests et d'autre part les résultats de ces tests.

Le benchmark utilisé pour les tests est bâti sur une plateforme de données basée sur deux sources de films : MovieLens [ML06] et IMDb [IMDb07]. Ce chapitre présente un résumé du processus de construction de la plateforme de données qui comprend : (i) l'extraction et l'intégration des données des deux sources et (ii) la construction de profils, de requêtes et de résultats de référence pour chaque couple (profil, requête). À partir de cette plateforme, un benchmark a été dérivé. Une description des principales étapes de la construction du benchmark est faite dans le chapitre. Elle comprend la simulation d'un système distribué et le choix d'un échantillon de requêtes et de profils.

Les évaluations des trois approches sont présentées à deux niveaux : niveau sémantique (compilation) et niveau exécution. Les résultats ont montré que l'introduction de la personnalisation dans la reformulation des requêtes a des bénéfices, mais que ces bénéfices ont un coût.

Le Chapitre 7 présente la conclusion de la thèse et les perspectives de recherche qu'elle ouvre.

Chapitre 2. État de l'art

Ce chapitre résume les grandes lignes de l'état de l'art sur la personnalisation. Les principaux points abordés sont les applications de personnalisation, les modèles de définition et de représentation des profils utilisateurs, les techniques de construction et d'exploitations des profils. Un état de l'art écrit en anglais qui détaille plus les points abordés dans ce chapitre peut être trouvé dans l'Annexe 5.

1. Introduction

Ce chapitre présente les grandes lignes de l'état de l'art sur les profils utilisateurs et sur les techniques de gestion et d'exploitation de ces profils.

Les systèmes d'information actuels donnent accès à un grand nombre de sources hétérogènes et distribuées. Au fur et à mesure que les sources se multiplient et que le volume de données disponibles s'accroît, les utilisateurs sont confrontés à une surcharge informationnelle dans laquelle les informations pertinentes se mélangent à des informations secondaires et même à du bruit. Par ailleurs, la même requête émise par deux utilisateurs dont les besoins et les intentions sont différents, produit les mêmes résultats. Pour répondre aux problèmes de la surcharge informationnelle et pour pouvoir discriminer les utilisateurs en fonction de leurs besoins spécifiques, de plus en plus de systèmes proposent des techniques de personnalisation. Parmi ces systèmes, on peut citer ceux qui permettent à l'utilisateur de personnaliser les rubriques et l'apparence de ses pages d'accueil (MyYahoo[2], iGoogle[3]), ceux qui font des recommandations aux utilisateurs en fonctions de l'historique de leurs actions (Amazon[4], dailymotion[5], etc.) ou encore les applications de vidéo à la demande [KKB+05] et d'IPTV [ARM+06].

La personnalisation des données a été abordée dans différents domaines comme la Recherche d'Information (RI), les Bases de Données (BD) ou l'Interaction Homme-Machine (IHM). Dans le domaine des IHM [EP00, LS01], la personnalisation se focalise principalement sur le niveau d'expertise de l'utilisateur afin de lui fournir les interfaces de communications et le type de dialogue les plus appropriés. Dans le domaine de la RI [CN04, BRS00, SL01, FS01, ZTB07], l'utilisateur fait partie du processus de personnalisation qui est vu comme un processus incrémental et interactif dans lequel l'utilisateur décide à chaque pas quels sont les éléments qu'il aime et quels sont ceux qu'il n'apprécie pas. La personnalisation est ainsi définie comme un apprentissage réalisé à partir des préférences rendues par les utilisateurs à l'issue de la présentation des résultats successifs du système. Dans le domaine des BD [Kie02, Kie05, KI04a, KI05b, Cho02, BKS01, LL87, RL06], l'utilisateur ne fait pas partie du processus de recherche d'informations. La requête contient en général l'ensemble des critères considérés nécessaires à produire des données pertinentes. La personnalisation de l'information se fait par l'extension des langages de requêtage (généralement SQL) ou par un processus de reformulation de requêtes qui utilise les profils utilisateurs.

[2] MyYahoo : http://cm.my.yahoo.com/
[3] iGoogle : http://www.google.fr/ig
[4] Amazon : http://www.amazon.com/
[5] Dailymotion : http://www.dailymotion.com

La personnalisation de l'information se fait généralement par la prise en compte d'un ensemble de connaissances, désignées souvent par les notions de *profil* et de *contexte*, qui décrivent les utilisateurs, leurs préférences et l'environnement dans lequel ils se trouvent. La section suivante présente les caractéristiques principales des profils utilisateurs existants.

2. Le profil utilisateur

Le profil utilisateur est une mise en facteur de la partie invariante des préférences de l'utilisateur. Même si l'ensemble des approches de personnalisation s'accordent sur l'importance de disposer d'un profil utilisateur, il manque un consensus sur la typologie des connaissances le constituant ainsi que sur la manière de représenter ces connaissances. Pour la majorité des approches, le profil contient la description du centre d'intérêt de l'utilisateur. Parmi les formalismes utilisés pour le représenter, on peut citer les vecteurs de mots clés (éventuellement pondérés) [CN04], les ontologies [Dem02], les fonctions d'utilités [CGFZ03], les ensembles de prédicats pondérés [KI04a, KI05b] ou encore les expressions logiques à la clause de Horn [KI04b, APV02, GL94].

Le contenu d'un profil utilisateur ne se limite pas aux préférences sur le contenu des données. Le profil utilisateur peut également contenir des données personnelles [Pot04], des préférences sur la qualité des données [BR02] et sur la sécurité [CLM02] ou encore des préférences sur le contenant [NFS98, VRM+06] et sur le processus de la production [BR02] et/ou de la livraison des données [GM05]. Dans ce cadre, la classification, l'organisation et la structuration des données du profil est un élément clé de la personnalisation. Différents travaux ont abordé cet aspect sans le couvrir dans son ensemble. Par exemple, P3P [CDE+05], standard pour la sécurisation des profils, permet de définir des classes distinguant entre les *attributs démographiques*, *les attributs professionnels* et *les attributs de comportement*. Dans [AS99], les auteurs proposent un modèle de profil pour les utilisateurs d'une bibliothèque digitale, composé de cinq catégories d'informations : *Données Personnelles*, *Données Collectées*, *Données de Livraison*, *Données de Comportement* et *Données de Sécurité*. Ces tentatives de structuration sont louables mais insuffisantes pour couvrir le champ de la personnalisation. Par ailleurs, elles se contentent de catégoriser les informations de profil, mais sont difficilement extensibles.

3. Les techniques de construction des profils utilisateur

Un problème important qui se pose dans le cadre des applications personnalisées est la construction des profils utilisateurs. Plusieurs travaux ont adressé ce problème dans différents domaines. De tels exemples sont les travaux de Mobasher et al. [Mob07, Mob05, AM05], qui présentent la personnalisation comme une application de techniques de data mining pour l'apprentissage automatique des profils, ou encore le travail présenté dans [HK06], qui propose une description de techniques de construction de profils dans le domaine des bases de données. La grande diversité des approches existantes nécessite l'identification d'un ensemble de paramètres qui permettant de les classer. Parmi les caractéristiques d'un processus de construction des profils, on peut citer :

- Le moment où le profil est construit: Généralement le processus de construction de profils utilise des techniques assez coûteuses basées par exemple sur l'apprentissage automatique [Cha00, EVV03, Fer95, SCZ05, JZM04] ou sur la théorie des graphes [Cha00, MC07, DDS02]. Par conséquent la construction des profils est souvent faite en dehors des sessions d'interaction entre l'utilisateur et le système. Parmi les approches qui proposent de construire les profils utilisateur en temps réel on peut citer les travaux de [PRGM03], [SHY04].

- L'implication de l'utilisateur : Selon ce critère, les approches se divisent en deux groupes : celle qui n'impliquent pas l'utilisateur dans la construction du profil [Cha00] et celles qui impliquent l'utilisateur en lui demandant de remplir un formulaire (Yahoo) ou de corriger manuellement ses préférences [MWT04].
- Le type de sources de données utilisées pour la construction des profils : Différentes types de données peuvent être exploitées pour la construction des profils utilisateurs. Des exemples sont les données d'usage [Mob05], les données de contenu [MC07], les données décrivant la structure et l'organisation des documents [Lie95] ou encore les données provenant de l'utilisateur [SHY04].
- La manière de mise à jour des profils : Lorsque de nouvelles données sont collectées sur l'utilisateur ou sur son comportement, le profil doit être mis à jour pour tenir compte de ces données. On distingue selon ce critère des systèmes qui mettent à jour uniquement les parties du profil utilisateur concernées par les nouvelles données et les approches qui construisent à nouveau le profil utilisateur.

Ces paramètres peuvent servir également comme critères de choix de la méthode de construction de profil la plus appropriée par rapport aux besoins des applications. On peut également remarquer qu'ils ne sont pas totalement indépendants les uns des autres. Par exemple si la construction du profil est faite en temps réel, alors il est difficilement envisageable de faire des mises à jour des profils par construction d'un nouveau profil.

4. Les techniques d'exploitation des profils utilisateur

Une fois le profil utilisateur construit, il est utilisé par les systèmes d'information afin de fournir à l'utilisateur des données pertinentes, dans le format le plus approprié, au bon moment et au bon endroit. Pour répondre aux besoins de la personnalisation, différentes approches ont été développées. Nous nous focaliserons ici sur les techniques de personnalisation utilisées dans le domaine des Bases de Données. Certaines approches proposant des algorithmes d'exploitation des profils dans le domaine de la Recherche d'Information peuvent être trouvées dans l'Annexe 5.

Depuis quelques années, les systèmes de bases de données proposent des langages de requêtes avancées qui supportent la notion de préférence. Il en est ainsi pour les langages comme PreferenceSQL [Kie02] ou SQL /f [RL06] ou pour les opérateurs de type « Best of » [Cho02], et « Skyline » [BKS01]. Ces travaux puisent leurs sources dans les recherches sur les bases de données floues [BP95, LYZ01] et l'exécution flexible de requêtes [PS04]. Bien que la notion de profil n'existe pas dans ces approches, elles ouvrent la voie à la personnalisation de l'accès aux données. Une partie du profil utilisateur (principalement le centre d'intérêt) peut être intégrée aux requêtes par un processus de réécriture et ainsi simplifier l'expression des requêtes par l'utilisateur.

Le travail mené dans [KI04a, KI05b] propose une approche d'enrichissement de la requête de l'utilisateur par de nouveaux critères de sélection contenus dans son profil. Dans cette méthode, 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. Il est spécifié par un nombre réel compris entre 0 et 1. L'enrichissement d'une requête à l'aide d'un profil consiste à ajouter dans la requête initiale les prédicats de sélection les plus pertinents. Cette approche s'inscrit dans le cadre de l'accès à une source de données unique. Pour tenir compte de l'aspect multi source, certaines approches ont proposé des techniques de réécriture de requêtes.

La réécriture de requêtes consiste à transformer la requête de l'utilisateur exprimée sur le schéma virtuel afin qu'elle puisse être évaluée sur les sources de données. Nous nous

limiterons à présenter la réécriture de requêtes dans un contexte LAV qui est le mieux adapté à une évolution rapide des sources de données.

La réécriture d'une requête dans une approche *Local As View* consiste à déterminer les sources contributives pour l'exécution de la requête utilisateur et à utiliser leurs définitions pour reformuler cette requête [CLL01]. Il existe deux classes principales d'algorithmes de réécriture: les algorithmes de règles inversées [DG97] et les algorithmes à base de tas (*Bucket-based*) [LRO96]. L'ensemble de ces algorithmes ne tiennent pas compte des préférences de l'utilisateur. Pour tenir compte des profils utilisateurs certaines approches utilisent des critères de qualité pour sélectionner les sources de données les plus pertinentes [NFS98] ou pour trouver les Top K chemins de navigation entre une source initiale et une source destination [VRM+06].

L'enrichissement et la réécriture sont des processus de reformulation de la requête utilisateur qui ont des objectifs différents. L'enrichissement permet de prendre en compte les préférences de l'utilisateur et de mieux cibler ses besoins tandis que la réécriture est faite dans le but d'accéder aux sources de données réelles. Ainsi il manque une approche d'accès personnalisé à des sources de données multiples qui combine les deux techniques.

5. Conclusion

Bien que la recherche sur la personnalisation ait produit un nombre important de résultats, que nous avons présenté dans ce chapitre, elle a soulevé un ensemble de questions et de problèmes qui restent ouverts.

Un problème clé de la personnalisation est la définition d'un modèle utilisateur. Les modèles actuellement utilisées par les systèmes personnalisés sont simplistes (ensembles de mots clés, prédicats pondérés, etc.) ce qui nécessite de mettre en place de modèles plus expressifs. Ces modèles doivent être multidimensionnels afin de représenter tous les aspects de l'utilisateur. En outre, il reste à définir un formalisme générique d'expression des préférences.

Un autre problème auquel doivent faire face les systèmes d'informations est l'aspect dynamique et évolutif du profil utilisateur. En effet, l'environnement dans lequel se trouve l'utilisateur peut influer sur le contenu du profil utilisateur. Dans ce cadre, il est indispensable de disposer d'un formalisme permettant de décrire et de représenter le contexte dans lequel se trouve l'utilisateur. Ainsi, un objectif important de la personnalisation est d'identifier les changements des préférences de l'utilisateur afin de proposer un moyen d'adaptation à ces changements.

Actuellement, chaque approche de personnalisation propose son propre framework et il manque un cadre général pouvant supporter différents types d'applications. Une tâche difficile est la mise en place de frameworks capables de réconcilier des applications ayant des besoins différents (personnalisation en temps réel, filtrage collaboratif, recommandation, etc.).

Une amélioration générale des systèmes de personnalisation serait de prendre en compte plus de sémantique dans leur processus. Cet aspect doit être pris en compte aussi bien dans le processus de construction que de celui de l'exploitation du profil utilisateur. Un moyen d'atteindre cet objectif est d'adapter certains résultats obtenus dans le domaine de la RI aux travaux des BDs. En effet, certains problèmes liés à la sémantique ont déjà été traités dans ce domaine.

Finalement, étant donné que le but principal de la personnalisation est de satisfaire au mieux l'utilisateur, il faut développer des métriques permettant de mesurer le degré de satisfaction. Les approches d'évaluation actuelles sont principalement basées sur l'exactitude

des résultats et le niveau de satisfaction de l'utilisateur n'est pas pris en compte. En outre, des métriques tenant compte du gain financier apporté par les techniques de personnalisation, permettrait de mesurer leur bénéfice économique.

Les contributions de cette thèse concernent (i) la modélisation des connaissances sur l'utilisateur et (ii) les techniques d'exploitation des profils et l'évaluation de ces techniques.

Chapitre 3. Meta modèles pour un système de personnalisation

Ce chapitre présente les méta modèles de profil, de contexte et de préférences ainsi que les relations qui existent entre eux. Il décrit également les principaux opérateurs de gestion de profils. Les méta modèles et les opérateurs sont implémentés dans une plateforme de gestion de profils et de contextes qui est également présentées dans ce chapitre.

1. Introduction

Le travail décrit dans cette thèse s'inscrit dans le cadre du projet APMD (Accès Personnalisé à des Masses de Données). Ce projet a pour objectif d'étudier de façon systématique et aussi générique que possible, d'une part, les différents types d'informations utilisées par les systèmes d'information pour rendre des résultats adaptés aux utilisateurs et, d'autre part, les techniques sous-jacentes à l'exploitation de ces données aussi bien dans un environnement de RI que de BD. Pour supporter cette étude, il est indispensable de disposer d'une infrastructure générique de définition et de gestion de profils. Dans ce cadre, la classification, l'organisation et la structuration des données décrivant l'utilisateur est un élément clé si on veut avoir une vision globale de la personnalisation.

La définition d'un profil peut se faire de deux façons différentes, éventuellement complémentaires: par instanciation d'un modèle générique de profils ou par spécification au moyen d'un langage de description de profils. La première approche est plus simple, elle consiste en une identification et une modélisation conceptuelle des différents attributs de profils. Elle a l'avantage d'une implémentation rapide mais a l'inconvénient d'être bornée par les seuls types d'informations prévus. La seconde approche est plus ouverte mais il est plus difficile de trouver un langage uniforme en raison de la grande diversité des informations constituant un profil. En tout état de cause, même l'approche langage nécessite une identification des types d'information constituant un profil; c'est la raison pour laquelle nous abordons la première approche en priorité dans ce chapitre.

Le profil d'un utilisateur ne contient pas uniquement des informations factuelles le décrivant. La notion de profil est souvent liée à celle de préférences et de contexte. En effet, les préférences de l'utilisateur font partie intégrale de son profil. En plus la description de l'utilisateur et ses préférences peuvent changer en fonction du contexte dans lequel il évolue. Cependant il y a une ambiguïté autour des trois concepts : profil, contexte et préférences. Le sens qu'on leur donne change d'une approche à l'autre et il arrive souvent que l'un d'entre eux soit utilisé à la place des deux autres ou des trois à la fois. Cette ambiguïté de la terminologie rend difficile l'étude et la compréhension de la problématique liée à la personnalisation.

L'objectif de ce chapitre est de clarifier ces concepts à travers une classification des connaissances qui les constituent, et d'établir un cadre de référence pour définir et manipuler des systèmes d'accès personnalisés à des masses de données. Ce cadre de référence se présente sous forme d'un modèle générique instanciable et adaptable à chaque type d'application. Pour atteindre cet objectif, les concepts de profil, de contexte et de préférence

sont analysés séparément. Ensuite ils sont intégrés à travers les relations sémantiques qui montrent clairement leur utilisation conjointe.

Ce chapitre est organisé comme suit. La section 2 présente les méta modèles de profil, de contexte et de préférences ainsi que les relations entre ces modèles. La section 3 discute les opérations définies sur les méta modèles. La section 4 présente la plateforme de gestion des méta modèles. Finalement la section 5 conclut le chapitre.

2. Méta modèles

Nous faisons une nette distinction entre les notions de profil, de préférence et de contexte. Un *profil utilisateur* représente l'ensemble des informations décrivant l'utilisateur. Un *contexte* représente les données décrivant l'environnement d'interaction entre un utilisateur et un système. Une *préférence* est une expression permettant de hiérarchiser l'importance des informations dans un profil ou un contexte.

L'objectif de cette section est de proposer des méta modèles génériques capables de décrire l'utilisateur, le contexte et les préférences. Le terme « générique » ne veut pas dire un ensemble exhaustif d'attributs, mais plutôt une liste de concepts de haut niveau pouvant être spécialisés, affinés et instanciés dans chaque environnement de personnalisation. Il y a toutefois une différence importante entre les méta modèles de profil et de contexte, d'une part, qui caractérisent les individus et les situations dans lesquelles ils interagissent avec un système d'information, et le méta modèle de préférences, d'autre part, qui décrit une typologie de préférences utilisées dans les deux premiers.

Les trois méta modèles doivent satisfaire les exigences suivantes :

- ils doivent être capables d'acquérir les principales catégories des connaissances utilisées dans les systèmes de personnalisation actuels,
- ils doivent être indépendants de tout système de gestion des données et de toute technologie,
- la spécialisation, la généralisation et l'instanciation de ces modèles doit être facile,
- ils doivent être ouverts, facilement extensibles à d'autres types de connaissances et d'autres types de préférences.

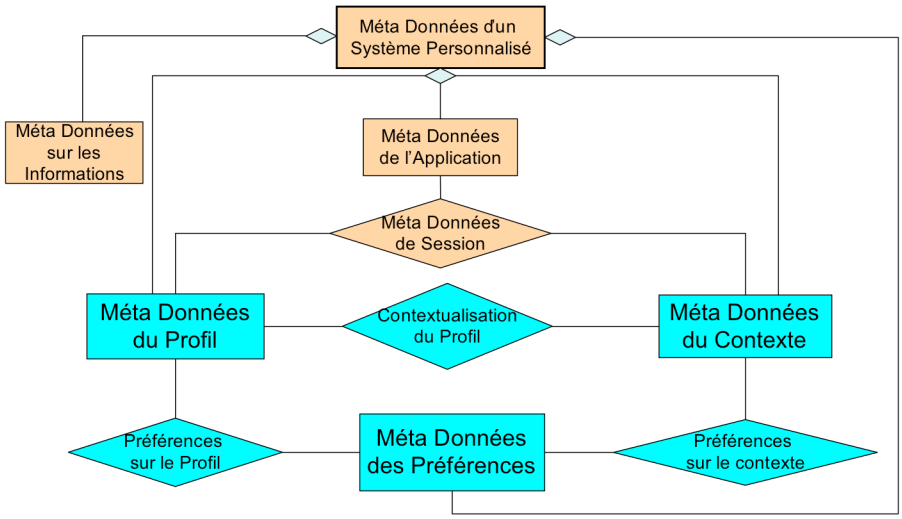


Figure 3.1 : Méta modèle global d'un système de personnalisation

Le méta données du profil, du contexte et des préférences font partie de l'ensemble de méta données d'un système personnalisé (Figure 3.1), mais ils ne constituent pas la totalité de

ces méta données. D'autres méta modèles décrivant le système d'information lui-même, en termes de données et de processus sont aussi nécessaires. Mais la description détaillée de ces derniers dépasse le cadre de cette thèse. Dans cette section, nous allons nous focaliser sur les méta modèles de profil, de contexte et de préférences ainsi que sur les relations existant entre eux.

2.1 Méta modèle de profil

Le profil utilisateur est composé de plusieurs dimensions (Figure 3.2). Chaque dimension est constituée d'un ensemble d'attributs éventuellement organisés en entités. Les attributs peuvent être simples ou composés. Les attributs composés sont appelés des sous dimensions. Une sous dimension regroupe un ensemble d'attributs simples qui sont liés sémantiquement (par exemple l'adresse est composée du numéro de la rue, du nom de la rue, du code postal etc.). Chaque attribut simple est caractérisé par son nom, le type de ses valeurs et la structure des valeurs. Le type d'un attribut peut être un des types couramment utilisés: entier, réel, chaîne de caractères, etc. La seconde caractéristique (la structure des valeurs) décrit le modèle de représentation des valeurs. Ce modèle de représentation peut être une valeur unique, un ensemble, un intervalle, un vecteur etc. À chaque attribut simple peuvent être associées une ou plusieurs valeurs.

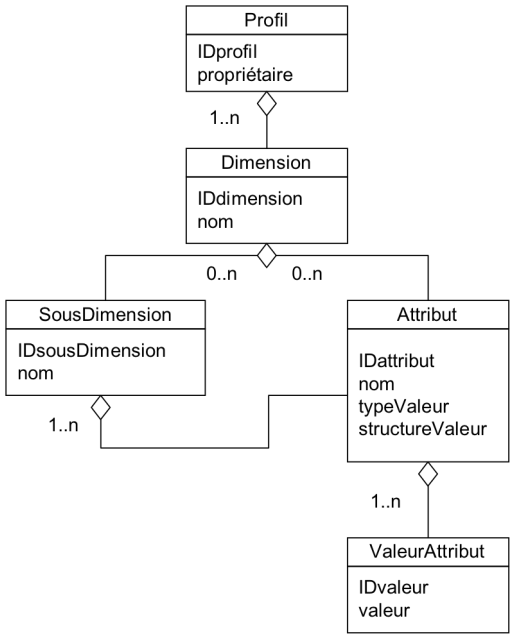


Figure 3.2 :Méta modèle de profil

Nous avons identifié 5 principales dimensions d'un profil utilisateur : domaine d'intérêt, données personnelles, données de qualité, données de livraison et données de sécurité. Les sections suivantes présentent le méta modèle de chacune de ces dimensions.

2.1.1 Domaine d'Intérêt

Le domaine d'intérêt est la dimension centrale du profil utilisateur. Cette dimension regroupe tous les attributs qui concernent les *objets de contenu* (informations ciblées). Elle peut définir aussi bien le domaine d'expertise et le niveau de qualification de l'utilisateur dans un domaine particulier que le contenu auquel s'intéresse l'utilisateur.

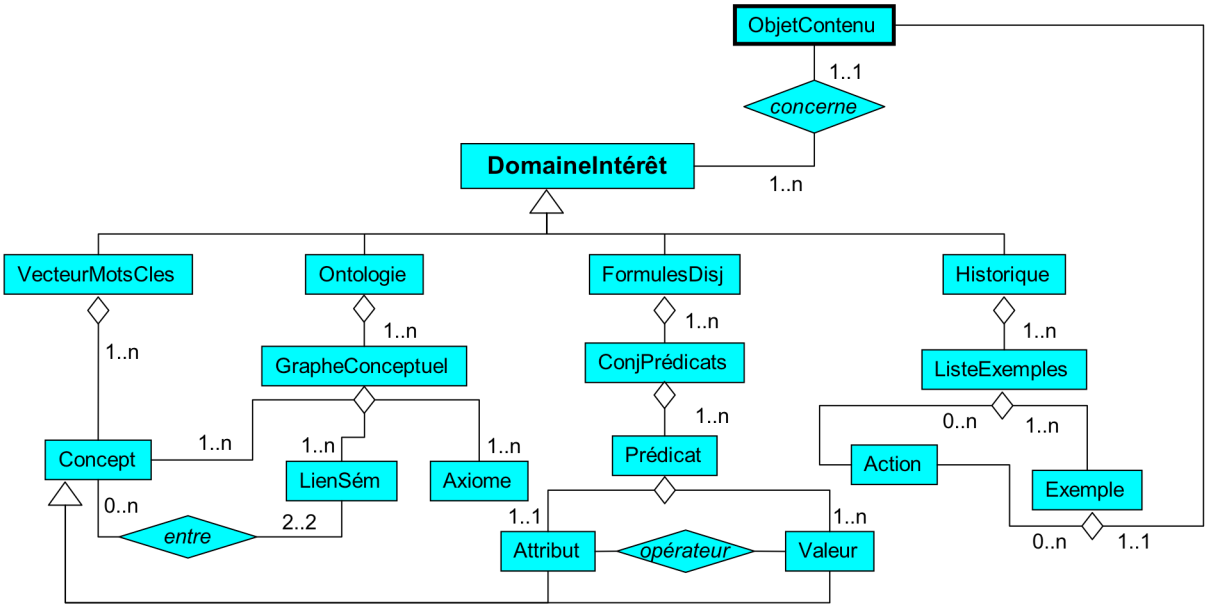


Figure 3.3 : Méta modèle de la dimension Domaine d'Intérêt

Le domaine d'intérêt peut être représenté de différentes manières (Figure 3.3) :

- Par un vecteur de mots clés, comme c'est généralement fait dans le domaine de la RI. Ces mots clés, éventuellement pondérés, traduisent les concepts auxquels l'utilisateur s'intéresse et que le système doit prendre en compte pour délivrer les documents qui les contiennent. Ces mots clés peuvent être organisés en graphes conceptuels ou ontologies explicitant les relations sémantiques entre les concepts.
- Par un ensemble de formules logiques, comme c'est généralement fait dans le domaine des bases de données. Le centre d'intérêt est alors représenté par un ensemble de formules disjonctives. Une formule disjonctive contient un ensemble de conjonctions de prédicats où chaque prédicat est défini sur un attribut d'une entité ou d'une table.
- Par un historique des interactions entre l'utilisateur et le système. Cet historique est représenté par des listes d'exemples annotés par les actions que l'utilisateur a effectuées sur un objet ou sur l'ensemble des objets (ex. lecture, envoie par mail, impression etc.).

2.1.2 Données personnelles

Les données personnelles sont la partie statique du profil. Elles contiennent des informations qui décrivent l'utilisateur et ne dépendent pas du système à interroger. Nous distinguons trois catégories de données personnelles : l'identité de l'utilisateur, les données démographiques et les contacts (Figure 3.4). La première catégorie (identité) est composée d'un ensemble d'attributs d'identification de l'utilisateur. Des exemples de tels attributs sont le nom et le prénom de l'utilisateur, son adresse, son numéro de téléphone ou de fax, son adresse email etc. La seconde catégorie de données personnelles contient un ensemble d'attributs démographiques comme par exemple la date de naissance de l'utilisateur, son genre, son revenu, son état civil, le nombre d'enfants etc. Finalement, les contacts de l'utilisateur représentent son carnet d'adresse. Un élément de ces contacts est représenté par les données personnelles d'un autre utilisateur.

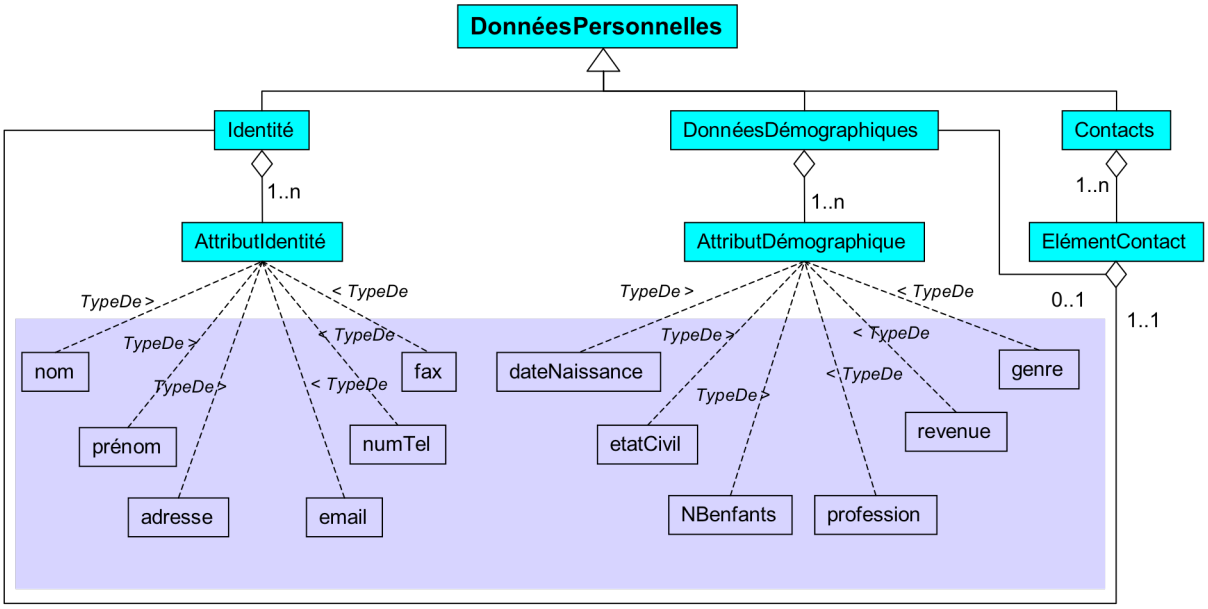


Figure 3.4 : Méta modèle de la dimension Données Personnelles

Les exemples d'attributs démographiques et ceux des attributs d'identité sont des exemples d'instanciation du méta modèle du profil. Sur la Figure 3.4 ces attributs sont représentés par des boîtes vides sur un fond bleu. Nous adoptons cette représentation des exemples d'instanciation pour le restant des méta modèles de ce chapitre.

Comme le montre la Figure 3.4, les catégories *identité* et *données démographiques* sont toutes les deux composées d'un ensemble d'attributs. La décision de séparer ces deux catégories vient du fait qu'elles contiennent des informations de différente nature. Par exemple l'accès à l'identité de l'utilisateur est généralement restreint. Souvent l'utilisateur ne souhaite pas révéler son identité dans le but de rester anonyme. Par conséquent, les attributs de cette catégorie doivent être bien protégées. Contrairement à son identité, l'utilisateur est souvent prédisposé à échanger ses données démographiques avec le système d'information contre des services de personnalisation. Un tel exemple représentent les systèmes de recommandation qui regroupent les utilisateurs selon leurs données démographiques. Le comportement des membres de chaque groupe peut alors être analysé afin d'identifier des objets dont le contenu est pertinent pour les participant du groupe. Ces objets sont ensuite recommandés aux utilisateurs du groupe qui ne les ont pas encore vus.

2.1.3 Qualité

La dimension « Qualité » joue un rôle très important dans le domaine de la personnalisation. Les données de cette dimension décrivent la qualité attendue ou espérée par l'utilisateur qui sera confrontée à la qualité effective produite par le système de recherche d'informations afin de restreindre l'espace de recherche. Les informations dans cette dimension sont représentées par des facteurs de qualité (Figure 3.5). Nous distinguons trois catégories de facteurs de qualité selon le type d'objets auxquels ils font référence : (i) facteurs sur le contenu des données, (ii) facteurs sur le contenant des données et (iii) facteurs sur les processus. La première catégorie concerne la qualité désirée des objets de contenu. Des exemples de facteurs de cette catégorie sont la fraîcheur et l'exactitude des données. La seconde catégorie regroupe les facteurs de qualité des conteneurs des données (sources). La plupart des facteurs de cette catégorie résultent de l'expérience des autres utilisateurs (ex. le niveau de confiance dans la source, la fiabilité etc.). Ces facteurs sont utilisés pour filtrer les sources de données lorsque leur nombre est trop important. Finalement, la troisième catégorie

de facteurs de qualité concernent les processus qui produisent, délivrent ou notifient l'arrivée des résultats. Les facteurs de cette catégorie mesurent aussi bien les performances des processus (ex. temps de réponse) que leur capacité de produire tous les résultats pertinents (rappel) et uniquement les résultats pertinents (précision).

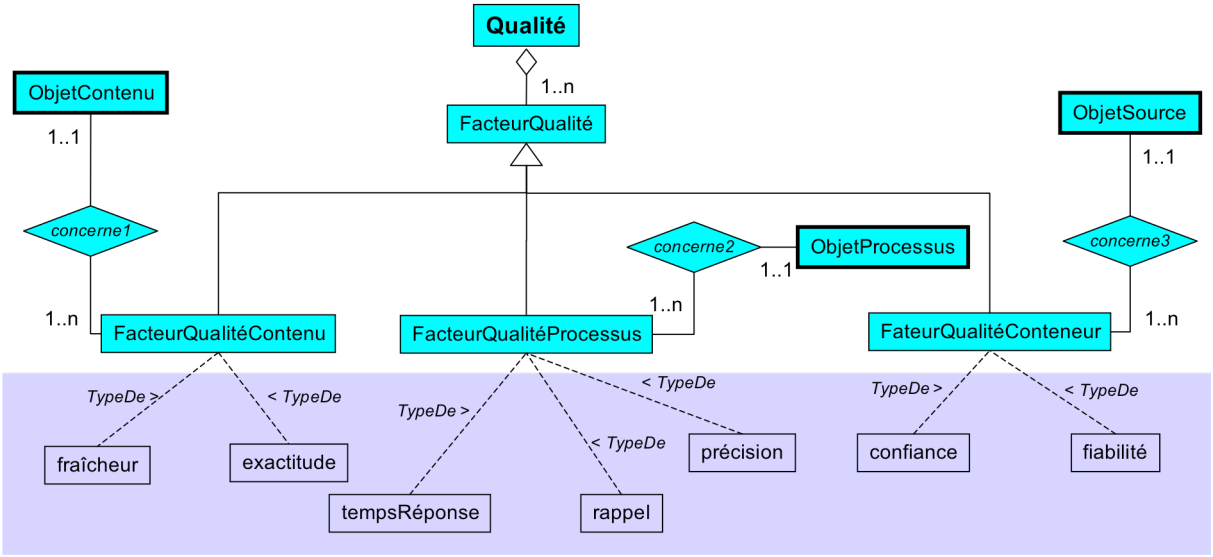


Figure 3.5 : Méta modèle de la dimension Qualité

2.1.4 Données de livraison

Les données de livraison concernent d'abord tout ce qui est lié aux modalités de présentation des résultats en fonction de la plateforme de l'utilisateur, de la nature et du volume des informations délivrées, des préférences esthétiques ou visuelles de l'utilisateur. À ces modalités de présentation, on peut ajouter les modalités de livraison et de notification, décrivant le moment et le moyen de livraison des résultats et la manière de notifier cette livraison (différé, immédiat par exemple).

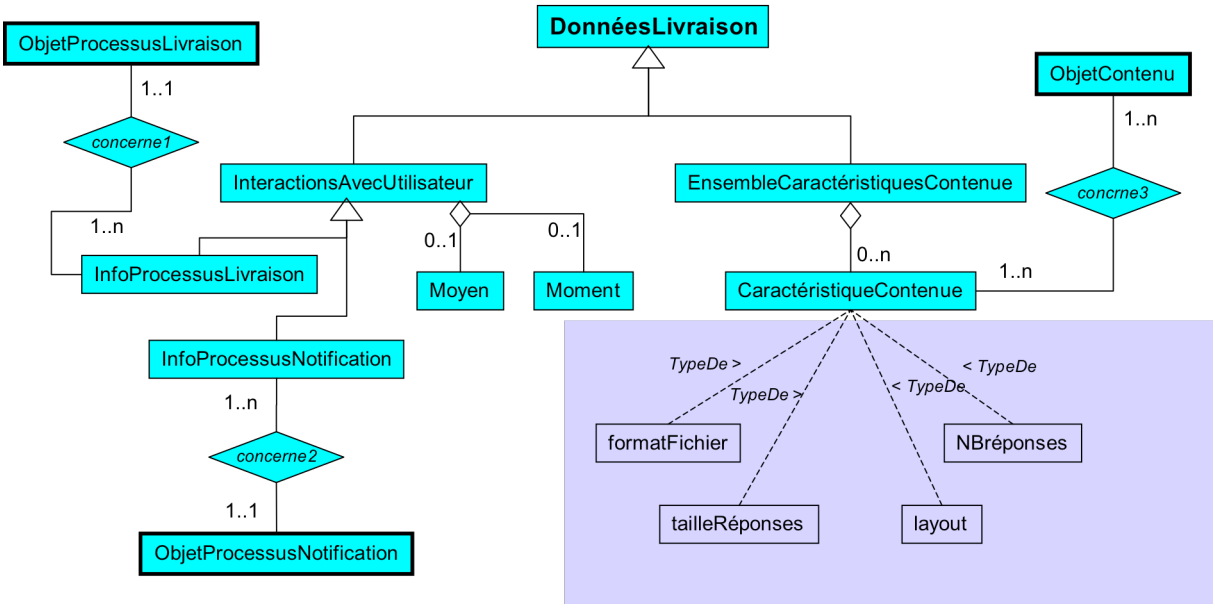


Figure 3.6 : Méta modèle de la dimension Données de Livraison

Les données de livraison sont divisées en deux classes : (i) des caractéristiques des objets de contenu délivrés et (ii) informations sur les interactions entre le système et l'utilisateur (Figure 3.6). La première classe regroupe l'ensemble de caractéristiques qui

concernent les objets de contenu. Ces caractéristiques comprennent par exemple le format des documents, leur nombre et leur taille ou encore la mise en page des données. La seconde classe fournit des informations sur le moment et le moyen des interactions entre l'utilisateur et le système. Ces interactions peuvent être la livraison des résultats ou la notification de leur arrivée. Dans le premier cas, les données de la dimension concernent le processus de livraison, alors que dans le second cas, elles caractérisent le processus de notification.

Il est important à mentionner que les données de livraison dépendent du contexte d'interaction. Par exemple, les résultats délivrés ne sont pas présentés de la même manière sur un laptop, sur un PDA ou sur un téléphone portable. Nous reviendrons sur cette dépendance plus tard dans ce chapitre.

2.1.5 Données de sécurité

La sécurité est une dimension fondamentale du profil. Elle peut concerner les données que l'on interroge ou modifie, les informations que l'on calcule, les requêtes utilisateurs ou les autres dimensions du profil.

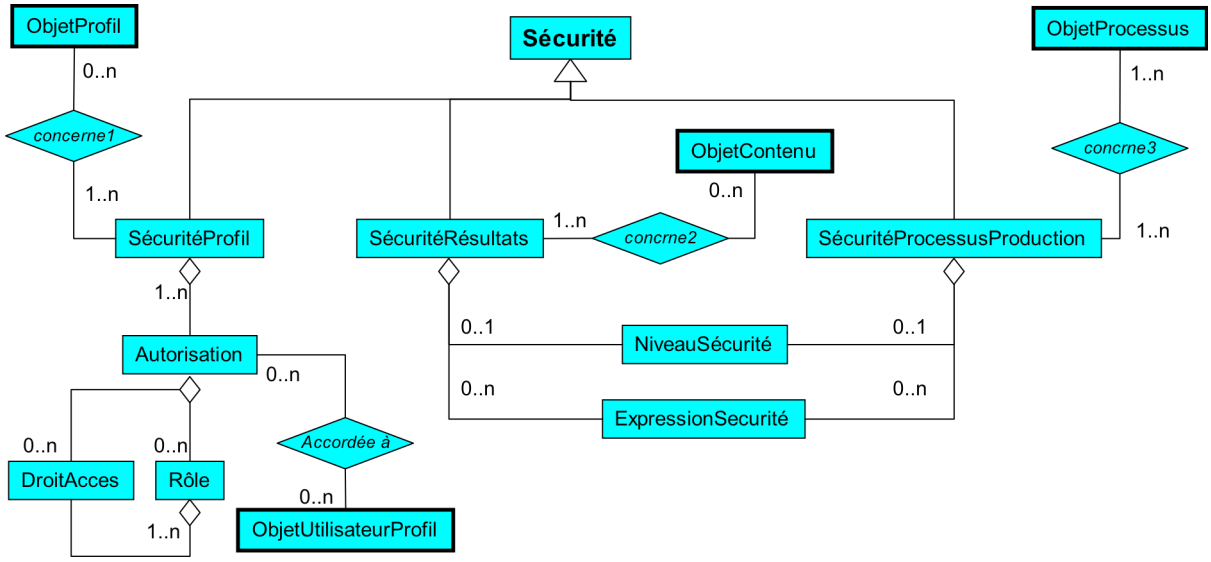


Figure 3.7 : Méta modèle de la dimension Sécurité

Nous distinguons trois types de sécurité en fonction des objets concernés : (i) la sécurité du profil utilisateur, (ii) la sécurité des résultats et (iii) la sécurité du processus de production des résultats (Figure 3.7).

Le premier type de sécurité (la sécurité des éléments du profil utilisateur) se fait à travers des autorisations d'accès aux informations du profil. Les utilisateurs d'un profil, qui peuvent être le système d'information ou les autres utilisateurs particuliers, se voient attribuer des droits d'accès ou des paquets de droits d'accès (rôle) afin de pouvoir interroger les données de ce profil.

La sécurité du second type (sécurité des résultats) concerne les objets de contenu qui sont délivrés à l'utilisateur. Elle est exprimée en indiquant le niveau de sécurisation des résultats (sur une échelle prédéfinie) ou à travers des expressions dans un formalisme existant. L'étude des différents formalismes d'expression de la sécurité dépasse le cadre de cette thèse.

La sécurité du troisième type (sécurité du processus) s'exprime de la même manière que celle de la sécurité des résultats. La sécurité de ce dernier type exprime la volonté de l'utilisateur de cacher un traitement sensible qu'il effectue. Par exemple, l'utilisateur d'un

système d'information proposant des informations sur la bourse peut vouloir cacher le fait qu'il fait des requêtes pour connaître le prix des actions des compagnies pétrolières.

Le méta modèle de profil présenté dans cette section n'est pas exclusivement réservé à la description de l'utilisateur. Il peut également décrire le système d'information ou les sources de données. Par exemple, le domaine d'intérêt d'une source décrit le contenu des informations qu'elle contient et les données de livraison décrivent entre autres le format de ces données. Toutes les dimensions ou tous les attributs des méta modèles précédents ne sont pas utiles à la description du système d'information ou des sources. Le Tableau 3.1, résume le contenu des profils du système et des sources. Pour chaque dimension du profil, ce tableau présente également les nuances de la typologie des données qu'elle peut contenir dans les profils du système ou des sources.

Dimension du profil	**Catégorie des données de la dimension**	**Profil Système**	**Profil Source**
Domaine Intérêt	Mots clés	Description des informations proposées par le système. Toutes les catégories du domaine d'intérêt sauf l'historique sont utilisables.	Description du contenu de la source. Toutes les catégories du domaine d'intérêt sauf l'historique sont utilisables.
	Ontologie		
	Formules disjonctives		
	Historique		
Données Personnelles	Identité	Identité du système (nom, URL, numéros de contact, ...).	Identité de la source (nom, URL, ...).
	Données démographiques	∅	∅
	Contacts	Données personnelles des sources que le système utilise.	∅
Qualité	Qualité du contenu	Qualité des informations proposées par le système (borne supérieure ou inférieure).	Qualité des informations proposées par la source.
	Qualité du conteneur	Niveau de qualité des sources du système (borne sup. ou inf.).	∅
	Qualité des processus	Qualité des processus du système qui produisent les données.	Qualité du processus qui permet d'interroger la source.
Données de livraison	Caractéristiques des données livrées	Caractéristiques possibles des informations livrées (ex. formats disponibles, taille des résultats, ...).	Caractéristiques des données dans la source (ex. format des données).
	Interaction	Médias et moments possibles de livraison ou de notifications des résultats.	Disponibilité de la source.
Sécurité	Sécurité du profil	Sécurité des données du profil du système.	Sécurité des informations décrivant la source.
	Sécurité des résultats	Niveaux de sécurité proposés par le système.	Niveau de sécurité proposé par la source.
	Sécurité des processus		

Tableau 3.1 : Contenu du profil d'un système et d'une source

Nous avons présenté dans cette section un méta modèle multidimensionnel de profil. Un profil contient des caractéristiques décrivant un utilisateur, un système d'information ou une source qui peuvent dépendre du contexte dans lequel ils sont exploités. La section suivante présente le méta modèle de contexte.

2.2 Méta modèle de contexte

Le contexte regroupe toutes les informations relatives à l'environnement dans lequel se fait l'interaction entre l'utilisateur et le système d'information. Pour le représenter, nous avons adopté la même démarche que pour la définition du profil utilisateur (voir Figure 3.2). Le contexte est composé d'un ensemble de dimensions où chaque dimension contient des attributs simples ou composés (sous dimensions). Les principales dimensions du contexte sont : la dimension spatiale, la dimension temporelle et l'équipement utilisé par l'interaction (Figure 3.8). Rappelons que le méta modèle de contexte, comme celui du profil, est ouvert et d'autres dimensions peuvent y être ajoutées si les besoins de l'application l'exigent.

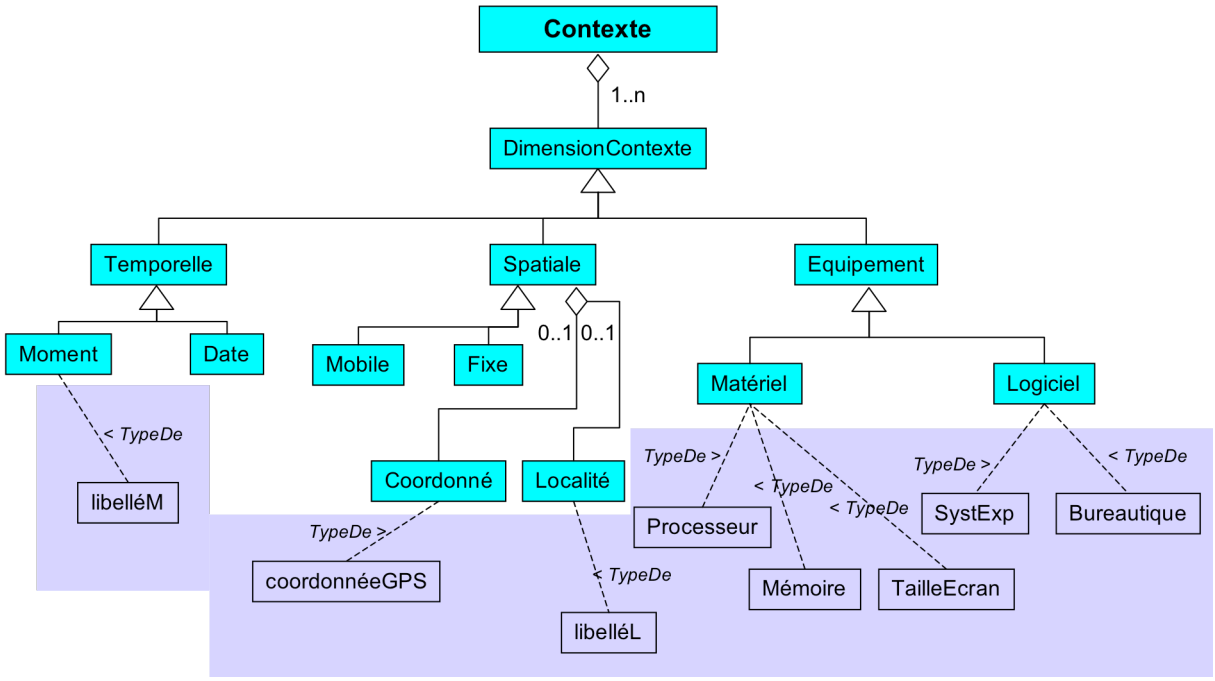


Figure 3.8 : Méta modèle du Contexte

Le contexte permet de situer l'utilisateur dans le temps et dans l'espace. Ceci est fait à travers les dimensions temporelle et spatiale. La dimension temporelle indique le moment de l'interaction. Ce moment peut être représenté dans un format de date standard ou en utilisant un libellé qui indique la période du temps (matin, soir, ...). Les libellés qui désignent le moment du contexte dépendent de l'application. Dans certains cas, « matin » peut désigner la période entre 6h00 et 11h00, et dans d'autres cas le même libellé peut indiquer la période entre 9h00 et 12h00.

La dimension spatiale désigne l'endroit dans lequel se trouve l'utilisateur. Cet endroit peut être statique (à la maison) ou dynamique (en voiture). La dimension spatiale peut contenir les coordonnées précises de l'utilisateur ou le libellé général de la localité dans laquelle il se trouve (travail, maison, voiture, ...). Comme pour la dimension temporelle, les libellés de la dimension spatiale peuvent changer d'une application à une autre.

En plus de l'aspect spatio-temporel, le contexte contient des informations sur l'équipement de l'utilisateur. Cet équipement peut être représenté par des caractéristiques du matériel et/ou du logiciel dont se sert l'utilisateur. Des exemples de caractéristiques du matériel sont la taille de l'écran, la capacité de la mémoire, la rapidité du processeur etc. En ce qui concerne le logiciel, il peut être décrit par le système d'exploitation de la machine de l'utilisateur ou encore les logiciels de bureautique qu'il utilise. Les caractéristiques de l'équipement servent à rendre les résultats exploitables sur la plateforme de l'utilisateur.

Il est intéressant à remarquer que le contenu du contexte est très proche de celui de la dimension « Données de livraison » du profil utilisateur. Même s'il existe une forte ressemblance entre les deux, le contexte et les données de livraison ne sont pas redondants, mais complémentaires. Le contexte détermine les valeurs du profil utilisateur qui peuvent être prises en compte, mais si plusieurs options sont possibles, le choix de celle qui sera prise est fait en fonction des préférences qui se trouvent dans le profil. Par exemple si plusieurs formats de fichiers sont compatibles avec le système de l'utilisateur, alors le profil décidera lequel choisir en fonction des préférences de l'utilisateur. Il est évident que le choix dicté par le profil peut être contextualisé i.e. changer d'un contexte à un autre.

2.3 Méta modèle de préférences

Le méta modèle de préférence décrit les types de préférence qu'on peut trouver dans les profils et dans les contextes. Ce modèle n'est pas instanciable séparément des deux autres. Il fournit une palette de préférences qui peuvent être utilisées pour hiérarchiser l'importance des connaissances définies dans les profils et les contextes.

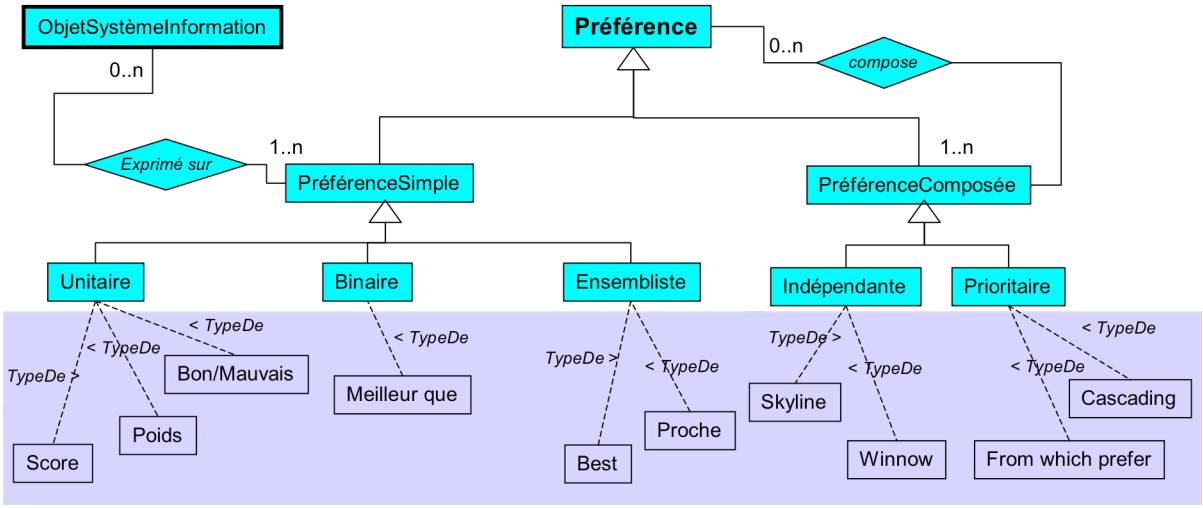


Figure 3.9 : Méta modèle de préférences

Il y a deux types de préférences : des préférences simples et des préférences composées (Figure 3.9).

Une préférence simple est définie par un seul concept caractérisant un ou plusieurs objets. Ce concept peut être un poids, un score, une fonction d'utilité, un prédicat ou tout mot clé introduisant une appréciation sur un objet ou un ordre d'intérêt entre deux ou plusieurs objets. Selon le nombre d'objets caractérisés, les préférences simples peuvent être unitaires, binaires ou ensemblistes. Les préférences unitaires s'appliquent sur un seul objet. Souvent elles sont représentées par une annotation de l'objet qui peut être un score, un poids ou une note de pertinence. Les préférences binaires, comme leur nom l'indique, montrent un favoritisme entre deux éléments. Elles sont souvent exprimées par des fonctions binaires de préférence permettant d'établir une relation d'ordre entre les éléments. Dans certains cas, la relation d'ordre peut exprimer le fait que les deux éléments sont incomparables i.e. il est impossible de choisir entre les deux. De tels éléments sont considérés comme ayant le même degré de pertinence. Finalement, les préférences ensemblistes s'appliquent à un ensemble d'objets. Elles permettent de distinguer le sous ensemble pertinent d'éléments. Des exemples de telles préférences sont la recherche des meilleurs objets (Best) ou de ceux qui s'approchent le plus d'un ensemble d'exemples pertinents (Proche). Les trois types de préférences simples ne sont pas complètement indépendants. La préférence unitaire est utilisée pour trouver le degré de pertinence d'un seul élément. La préférence binaire peut être établie en utilisant les

préférences unitaires de deux éléments. Plusieurs préférences binaires forment un ordre entre un ensemble d'éléments et cet ordre peut être utilisé par une préférence ensembliste pour sélectionner le sous ensemble pertinent.

Une préférence composée est une expression de deux ou plusieurs *préférences intermédiaires*. Une préférence intermédiaire peut être une préférence simple ou une préférence composée. La combinaison de préférences peut se faire de façon indépendante ou prioritaire. Lorsqu'elles sont combinées de façon indépendante, les préférences intermédiaires ont la même importance. En appliquant une telle préférence, il est important de satisfaire le plus grand nombre de préférences intermédiaires sans se soucier de l'ordre dans lequel elles sont appliquées. Si l'ordre dans lequel les préférences sont appliquées est important, alors elles sont combinées de façon prioritaire. Les préférences prioritaires forment une hiérarchie dans laquelle les préférences des niveaux supérieurs sont considérées comme étant plus importantes que celles des niveaux inférieurs. Dans ce cas, les préférences peuvent être vues comme des filtres qui sont appliqués dans l'ordre établi par la hiérarchie. Étant donné que la composition hiérarchique se fait sur des préférences intermédiaires, il est possible d'obtenir toutes sortes de compositions de préférences simples. Il est possible par exemple d'avoir des préférences simples qui ont la même importance et d'autres qui sont moins importantes dans la même préférence composée.

2.4 Relations entre le profil, le contexte et les préférences

Dans les sections précédentes, la description de l'utilisateur, la représentation du contexte et les expressions des préférences ont été séparées afin de mieux les étudier. Cependant, une représentation complète de l'utilisateur et du contexte, contient non seulement des éléments de description, mais également des préférences. En plus, la définition d'un profil utilisateur peut dépendre du contexte dans lequel il est exploité. Par conséquent, l'obtention d'un modèle complet décrivant l'utilisateur et/ou le contexte passe obligatoirement par l'identification des relations qui existent entre les trois composants profil, contexte et préférences.

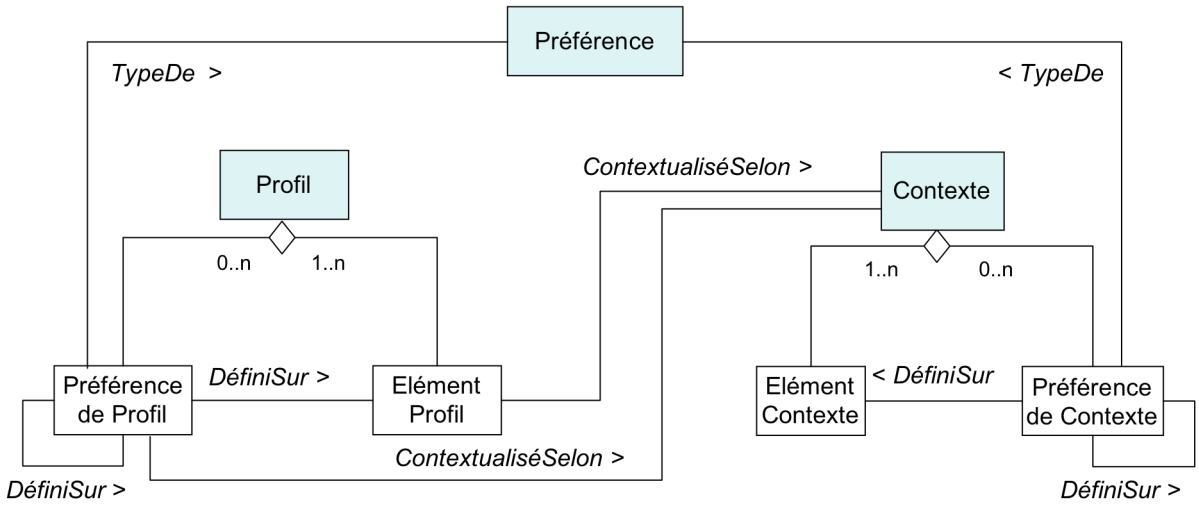


Figure 3.10 : Associations entre profil, contexte et préférences

La Figure 3.10 illustre les principaux liens qui existent entre le profil, le contexte et les préférences. En plus de l'agrégation, trois catégories d'associations sont utilisées pour construire le modèle de cette figure : « *type de* », « *défini sur* » et « *contextualisé selon* ». Les liens de la première catégorie permettent de différencier les préférences du profil de celles du contexte. La seconde catégorie de liens sert à relier les préférences aux éléments sur lesquels

elles sont exprimées. Finalement, les associations « *contextualisé selon »* assurent la relation entre le contexte et le contenu du profil.

Un profil utilisateur est composé d'un ensemble d'éléments de profil et de préférences. Les éléments du profil peuvent être des dimensions, des sous dimensions, des attributs et des valeurs. Des préférences peuvent être définies sur l'ensemble de ces éléments à condition de combiner des éléments du même type. Par exemple il est possible d'exprimer une préférence entre deux valeurs du même attribut, mais il est difficile d'envisager une préférence entre une dimension du profil et la valeur d'un attribut.

Le contexte est défini de la même manière que le profil. Il est composé d'un ensemble d'éléments de contexte et de préférences. L'utilisation de préférences au niveau du contexte paraît moins naturelle que celle au niveau du profil. Les préférences du contexte permettent de définir un choix par défaut sur les caractéristiques du contexte. Par exemple, il est possible d'exprimer le fait que l'ordinateur à partir duquel l'utilisateur se connecte est plus important que son emplacement physique pour décider s'il est dans un contexte de travail ou de loisir.

En plus d'être définies sur les éléments du profil et du contexte, les préférences peuvent être définies sur d'autres préférences pour former des expressions plus complexes (voir section 2.3). Il y a donc une association réflexive du type « *défini sur »* sur les préférences du profil et sur celles du contexte.

Finalement, le contenu du profil peut dépendre du contexte. Cette dépendance est exprimée par les associations « *contextualisé selon »*. Un élément du profil ou une préférence peut être contextualisé ou non. Lorsqu'un composant (préférence ou élément du profil) est non contextualisé, il est utilisé systématiquement dans tous les contextes. Un exemple de tel composant est le nom de l'utilisateur qui ne change jamais.

Contextualiser un composant revient à rendre son contenu dépendant de l'environnement dans lequel il est exploité. La contextualisation concerne aussi bien la description de l'utilisateur que ses préférences. Pour qu'un élément du profil puisse être contextualisé, il doit contenir plusieurs valeurs. La contextualisation est alors faite en spécifiant quelles sont les valeurs à pendre en compte selon le contexte. Par exemple il est possible d'avoir une adresse email pour le contexte « Loisir » et une autre pour le contexte « Travail ». De la même manière, les préférences du profil peuvent changer d'un contexte à un autre. Par exemple, l'utilisateur peut préférer recevoir des fichiers au format Postscript plutôt que des fichiers en PDF lorsqu'il est « à la maison », alors qu'« au travail » cette préférence peut être inversée (le format PDF est préféré devant le format PS).

Les modèles de profil, de contexte et de préférences, décrit dans cette section, servent à créer des profils. La section suivante présente le processus d'instanciation de profils et les opérations qui permettent de les manipuler.

3. Gestion de profils

La gestion de profils implique la mise en place d'un ensemble d'outils qui facilitent la définition et l'évolution des profils. Les principales opérations de gestion de profils sont :

- Instanciation : cette opération permet de créer des profils particuliers,
- Importation : cette opération permet de rendre un profil utilisable sur un système qui n'est pas celui qui a servi à le créer. Elle implique un test de validité du profil par rapport au modèle de profil reconnu par le système (validité sémantique),

- Validation : la validation d'un profil permet de déterminer quelle partie de ce profil peut être satisfaite sur un système particulier. Elle implique un test par rapport à l'environnement technique dans lequel le profil sera utilisé (validité opérationnelle),
- Intégration : l'intégration permet de combiner le contenu de deux ou plusieurs profils dans un seul. Cette opération peut être appliquée sur les différents profils du même utilisateur ou sur les profils de différents utilisateurs. Dans le premier cas, l'objectif est d'uniformiser la représentation de l'utilisateur, alors que dans le second cas il s'agit de créer le profil d'une communauté d'utilisateurs,
- Adaptation : l'adaptation d'un profil se fait toujours par rapport à un contexte. Elle permet d'obtenir le contenu du profil par rapport au contexte dans lequel se trouve l'utilisateur,
- Appariement : l'appariement de profils comprend un ensemble d'opérations de base qui sont le matching, la différence et l'équivalence de profils. Ce sont des opérations qui peuvent être utilisées comme briques de base pour obtenir les autres opérations.

La description détaillée de toutes les opérations est au-delà des limites de cette thèse. Nous allons nous focaliser sur l'instanciation et sur l'appariement des profils en raison de leur caractère fondamental pour les autres opérations. Nous allons également montrer comment certaines des opérations restantes peuvent être obtenues à partir des opérations d'appariement.

3.1 Instanciation des modèles

L'instanciation est un processus qui permet de créer des instances des différents modèles et d'adapter leur contenu à des besoins particuliers. Cette section présente les différents niveaux d'instanciation des profils, des contextes et des préférences et illustre ce processus par un exemple.

3.1.1 Niveaux d'instanciation

Il y a trois niveaux de représentation d'un profil, d'un contexte et d'une préférence : le niveau méta modèle, le niveau modèle et le niveau instance (Figure 3.11).

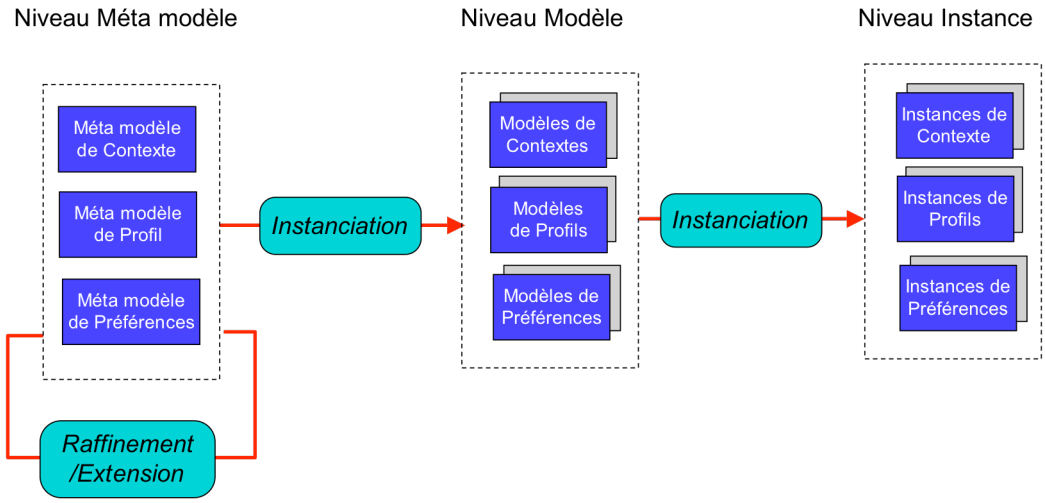


Figure 3.11 : Niveau d'instanciation

Le premier niveau est composé des méta modèles présentés dans la section 2. Comme nous l'avons mentionné précédemment, les éléments de ce niveau sont des modèles ouverts et extensibles. En cas de besoin, ils peuvent être raffinés ou étendues afin de correspondre aux besoins d'une classe d'applications.

Le second niveau de représentation correspond aux modèles types de profil, de contexte et de préférences. Ces modèles servent à satisfaire les besoins spécifiques d'un utilisateur et/ou d'une application.

Finalement, le troisième niveau de représentation comprend les instances particulières de profils, de contextes et de préférences pour une application donnée.

Le passage entre les trois niveaux de représentation se fait par l'opération d'instanciation. Par conséquent, l'obtention d'une instance d'un modèle, passe par deux niveaux d'instanciation : instanciation des méta modèles et instanciation des modèles.

Instanciation d'un méta modèle

L'instanciation d'un méta modèle se fait en choisissant les éléments de ce méta modèle dont a besoin l'utilisateur ou l'application. Cette étape de l'instanciation sert à obtenir des modèles types de profil, de contexte et de préférences. Pour un profil utilisateur par exemple, cette phase revient à choisir les dimensions, les sous dimensions et les attributs qui vont composer ce profil. L'idée derrière la définition de modèle types est de pouvoir les réutiliser autant de fois que nécessaire. Il est possible par exemple de créer un seul modèle de représentation du contexte dans une application et utiliser ce modèle pour créer tous les contextes dans lesquels peut se trouver un utilisateur de cette application.

Certains méta modèles de la section 2 contiennent des éléments du niveau modèle. Ces instances sont représentées par des boîtes vides sur un fond bleu et sont reliées aux éléments qu'elles instancient par une association « *type de* ». Par exemple, le *format* et le *nombre de réponses* sont des instances des *caractéristiques de contenu* du méta modèle de la dimension *données de livraison* (Figure 3.6).

Instanciation d'un modèle

Le second niveau d'instanciation permet d'obtenir une instance de profil, de contexte ou de préférence à partir d'un modèle particulier. Ceci est fait en deux phases : (i) choix du modèle type et (ii) attribution de valeurs aux attributs de ce modèle. La première phase consiste à trouver le modèle type qui correspond le mieux aux besoins de l'utilisateur et/ou de l'application. La seconde phase revient à attribuer les valeurs caractérisant l'utilisateur ou le contexte aux attributs du modèle identifié.

L'instanciation de modèles ne se résume pas uniquement à l'attribution de valeurs aux attributs de ce modèle. Comme nous l'avons montré dans la section 2.4 , il existe des relations entre le profil, le contexte et les préférences. Par conséquent l'instanciation de leurs modèles doit se faire en tenant compte de ces relations. Pour pouvoir le faire, les règles suivantes sont introduites :

- l'instanciation d'un modèle de préférence ne peut se faire que si l'élément sur lequel elle est définie existe déjà,
- pour pouvoir exprimer une préférence sur une instance d'un modèle de profil ou de contexte il faut instancier cette préférence,
- pour contextualiser un élément d'une instance de profil, il faut instancier les contextes par rapport auxquels cet élément sera contextualisé.

La première règle détermine une condition obligatoire pour l'instanciation d'une préférence. En effet, une instance de préférence est toujours définie sur un élément donné. Par conséquent, l'instanciation d'un modèle de préférence ne peut se faire que si l'élément sur lequel elle est définie existe déjà.

Les deux règles suivantes ne sont pas bloquantes dans le sens où elle n'empêchent pas l'instanciation d'un profil ou d'un contexte. La contextualisation et l'ajout de préférences sur les instances peuvent être faits en différé par rapport à l'instanciation. Le rôle de ces règles est de montrer que l'instanciation d'un modèle peut engendrer l'instanciation d'autres modèles. Par exemple, si on veut obtenir une instance d'un modèle de profil qui contient des éléments contextualisés, on doit instancier les contextes dont dépendent ces éléments.

Cette section a décrit le processus d'instanciation d'un profil, d'un contexte et d'une préférence. Pour mieux illustrer ce processus, la section suivante présente un exemple d'instanciation.

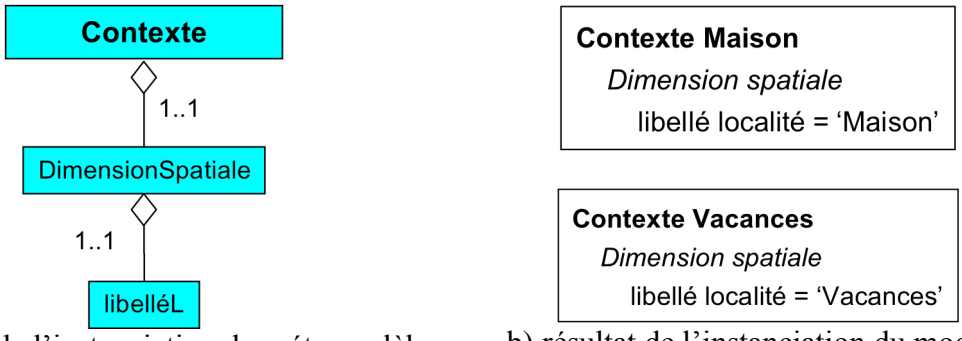
3.1.2 Exemple d'instanciation

Supposons que nous voulons représenter sous forme d'instance le profil d'un utilisateur qui possède les caractéristiques suivantes :

- il s'appelle *Victor Laporte*,
- il aime *recevoir* des annonces de films *après 20h*.,
- il souhaite que la *fiabilité des sources* desquelles proviennent les annonces soit *supérieure à 80%,*
- il *préfère* les annonces au *format PDF* par rapport à celle au format *PS* et en dernier recours peut lire des fichier *DOC*,
- lorsqu'il est *à la maison*, il aime recevoir les *dernières* annonces (celles de la journée) de films *d'action*, adore les films *réalisés par Coppola* et parfois regarde des *dessins animés*,
- lorsqu'il est *en vacances*, il souhaite regarder uniquement des *dessins animés*, étant donné qu'il part avec ses enfants qui sont petits. Pour trouver ces films il consulte les annonces *des 2 dernières années*.

Cette description de l'utilisateur contient des éléments qui dépendent du contexte (seuil ou en famille) et des préférences. Certaines préférences sont binaires (ex. le format PDF est préféré par rapport à PS) et d'autres sont unitaires (ex. aime recevoir des films d'action). Pour pouvoir instancier le profil, il faut donc instancier les contextes et déterminer les formalismes d'expression des préférences.

Instanciation des contextes



a) résultat de l'instanciation du méta modèle b) résultat de l'instanciation du modèle

Figure 3.12 : Instanciation des contextes

La seule caractéristique qui décrit les contextes est la localité de l'utilisateur qui est une caractéristique de la dimension spatiale. La localité et représentée par un libellé ce qui implique que l'instanciation du méta modèle du contexte permet d'obtenir un modèle qui ne

contient que l'attribut « libelléL » (Figure 3.12 a). Le même modèle peut alors être instancié deux fois pour obtenir les deux contextes « Maison » et « Vacances » (Figure 3.12 b).

Instanciation du profil

Les informations décrivant l'utilisateur peuvent être classées dans les dimensions du profil da la manière suivante :

- Le nom et le prénom de l'utilisateur font partie des données personnelles,
- la description des films qu'il recherche fait partie du domaine d'intérêt. Elle peut être faite par un ensemble de prédicats,
- le fait qu'il veut recevoir les annonces après 20h et le format des fichiers font partie des données de livraison,
- la fiabilité des sources et la fraîcheur des annonces sont des facteurs de qualité.

Pour pouvoir stocker ces informations, le méta modèle du profil est instancié. Le résultat de cette instanciation est le modèle type de la Figure 3.13.

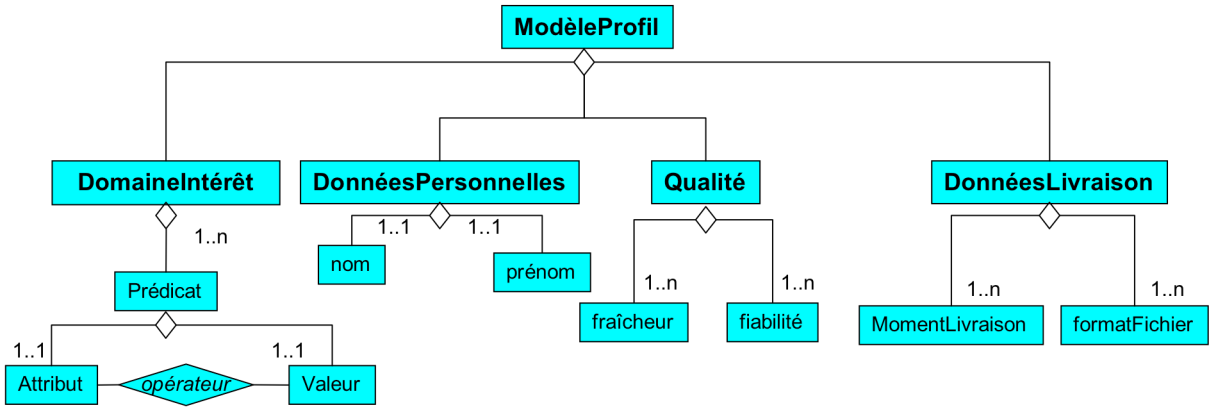


Figure 3.13: Résultat de l'instanciation du méta modèle du profil

La seconde phase de l'instanciation du profil est l'instanciation du modèle de la Figure 3.13. Le résultat de cette phase est illustré sur la Figure 3.14. L'instance de profil de cette figure tient compte des contextes. Les prédicats du domaine d'intérêt ne sont pas les mêmes dans les deux contextes et la fraîcheur exigée par l'utilisateur change d'un contexte à l'autre.

Profil de Victor

Domaine d'intérêt

Prédicat

= { Film.genre = 'action'
Film.genre = 'dessin animé'
Film.producteur = 'Coppola' }
si Contexte Maison

= { Film.genre = 'dessin animé' }
si Contexte Vacances

Données personnelles

Prénom = Victor

Nom = Laporte

Qualité

fiabilité sources > 80%

fraîcheur

< 1 jour *si Contexte Maison*

< 2 ans *si Contexte Vacances*

Données de livraison

moment de livraison > 20h

format des fichiers = {PDF, PS, DOC}

Figure 3.14 : Résultat de l'instanciation du modèle de profil utilisateur en fonction des contextes

Instanciation de préférences

Pour être complète, l'instance de profil doit être enrichie avec les préférences de l'utilisateur. Supposons que l'utilisateur veut exprimer ses préférences sur les prédicats du domaine d'intérêt sous la forme de poids compris dans l'intervalle [0, 1] et que la préférence d'un format de fichiers *format1* par rapport à un autre format *format2* est notée *format1>>format2*. Le modèle qui sert à instancier les préférences du profil de Victor ne contient que deux types de préférences (Figure 3.15). Il peut servir à créer plusieurs instances de préférences qui sont exprimées sur les éléments du profil. Le résultat de l'ajout des préférences au profil est illustré sur la Figure 3.16.

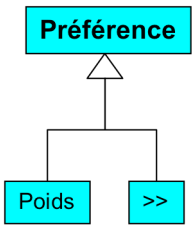


Figure 3.15: Modèle de préférences utilisées pour le profil de Victor

Profil de Victor
Domaine d'intérêt
Prédicat
= { poids(Film.genre = 'action') = 0.8
poids(Film.genre = 'dessin animé') = 0.2
poids(Film.producteur = 'Coppola') = 0.9 }
si Contexte Maison
= { poids(Film.genre = 'dessin animé') = 1.0 }
si Contexte Vacances
Données personnelles
Prénom = Victor
Nom = Laporte
Qualité
fiabilité sources > 80%
fraîcheur
< 1 jour *si Contexte Maison*
< 2 ans *si Contexte Vacances*
Données de livraison
moment de livraison > 20h
format des fichiers = {PDF >> PS >> DOC}

Figure 3.16 Résultat complet de l'instanciation du profil de Victor

Plusieurs opérations peuvent être définies sur les profils. La section suivante présente l'appariement de profils qui propose des primitives pouvant servir à construire les autres opérations.

3.2 Appariement de profils

La seconde opération que nous allons présenter est l'appariement de profils. Elle est constituée de primitives qui permettent de comparer deux profils pour s'assurer de leur équivalence stricte, isoler leurs parties communes ou énumérer leurs différences. Ces primitives s'appliquent aussi bien au niveau modèle des profils qu'à leurs valeurs. Dans le premier cas, elles permettent de comparer la structure des profils qui est la représentation des profils au niveau modèle, tandis que dans le second cas, les primitives permettent de confronter les valeurs des attributs de deux profils. Les trois primitives que nous avons identifiées sont : l'équivalence (*Equivalence(P_1,P_2)*), le matching (*Match(P_1,P_2)*) et la différence (*MisMatch(P_1,P_2)*). Pour expliquer le fonctionnement de ces primitives, nous allons utiliser la notation suivante :

- P_i représente une instance i de profil utilisateur,
- *Model(P_i)* est le modèle qui a servi à instancier le profil i. C'est l'expression du profil i au niveau modèle ce qui correspond à la structure arborescente des dimensions, des sous dimensions et des attributs du profil,
- *Value(P_i)* est la valeur du profil i. La valeur d'un profil correspond uniquement aux données de description le composant. Dans *Value(P_i)*, les valeurs de chaque attribut correspondent à l'union des valeurs de cet attribut dans l'ensemble des contextes et les préférences ne sont pas prises en compte. Autrement dit, la valeur d'un profil correspond à l'ensemble de valeurs du niveau instance associées aux attributs du niveau modèle.
- *Attributes(P_i)* est l'ensemble d'attributs (simples et composées) du profil P_i,

- *Dimensions(P_i)* sont les dimensions du profil P_i et
- *Values(A_j, P_i)* désigne les valeurs d'un attribut A_j dans le profil P_i.

Pour pouvoir illustrer l'appariement à travers un exemple, nous allons utiliser le profil de Victor (P_V) de la Figure 3.16 et un second profil que nous allons appeler le profil du Médiateur (P_M). Leur représentation au niveau modèle et au niveau valeurs est donnée sur la Figure 3.17.

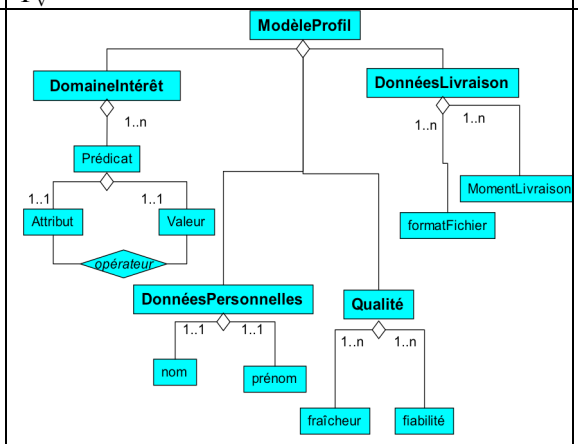
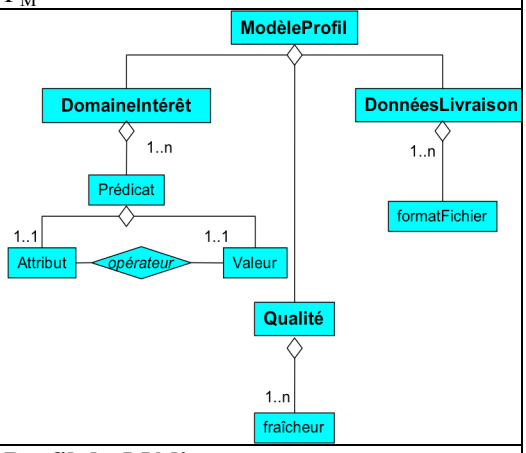
<table>
<tr><th></th><th>P_V</th><th>P_M</th></tr>
<tr><td>Niveau modèle (Model)</td><td> 
</td><td> 
</td></tr>
<tr><td>Niveau instance (Value)</td><td>Profil de Victor
Domaine d'intérêt
Prédicat = {
Film.genre = 'action'
Film.genre = 'dessin animé'
Film.producteur = 'Coppola' }
Données personnelles
Prénom = Victor
Nom = Laporte
Qualité
fiabilité sources > 80%
fraîcheur < 1 jour ∨ < 2 ans
Données de livraison
moment de livraison = après 20h
format des fichiers = {
PDF, PS, DOC }</td><td>Profil du Médiateur
Domaine d'intérêt
Prédicat = {
Film.genre = 'action'
Film.genre = 'dessin animé'
Film.genre = 'triller' }
Qualité
fraîcheur < 1/2 jour
Données de livraison
format des fichiers = {
PDF, PS, XML, HTML }</td></tr>
</table>

Figure 3.17 : Exemples de profils

3.2.1 Équivalence de profils

L'équivalence est une fonction booléenne. Appliquée au niveau modèle, elle permet de déterminer si deux instances de profils sont obtenues par instanciation du même modèle :

Définition 3.1 : Deux profils sont équivalents au niveau modèle s'ils sont obtenus par instanciation du même modèle.
Soient P_1 et P_2 deux profils.
Equivalence(Model(P_1), Model(P_2)) = VRAI si Model(P_1) ≡ Model(P_2)
FAUX sinon

Une utilisation possible de cette primitive est pour vérifier si un profil respecte le modèle imposé par une application.

L'équivalence au niveau valeur a pour objectif d'établir s'il y a une correspondance exacte entre deux profils aussi bien au niveau de leur structure (modèle) qu'au niveau des valeurs des attributs. Pour être équivalents au niveau valeur, deux profils doivent être obtenus par instanciation du même modèle et tous les attributs doivent avoir exactement les mêmes valeurs dans les deux profils :

Définition 3.2 : Deux profils sont équivalents au niveau valeur s'ils sont équivalents au niveau modèle et si tous les attributs ont deux à deux les mêmes valeurs dans les deux profils.
Soient P_1 et P_2 deux profils.

Equivalence(Value(P_1), Value(P_2)) = VRAI si Equivalence(Model(P_1), Model(P_2)) $\wedge \forall$ a $\in$ Attributes(P_1), a $\in$ Attributes(P_2) $\wedge$ Values(a, P_1) $\equiv$ Values(a, P_2)

FAUX sinon

Cette primitive peut servir à déterminer s'il y a des utilisateurs qui possèdent la même description afin de former des groupes d'utilisateurs.

Le résultat du test de l'équivalence des deux profils P_V et P_M de la Figure 3.17 est faux aux deux niveaux (modèle et valeurs). En effet, étant donné que les deux profils sot instanciés à partir de modèles différents, ils ne peuvent pas être équivalents.

3.2.2 Matching de profils

La seconde opération d'appariement est le matching. Le résultat du matching des modèles de deux profils est un modèle type qui est égal à leur intersection. Lors de sa construction il faut éviter les dimensions sans attributs. Même si une dimension apparaît dans les deux modèles de profil, elle ne fait partie du résultat que si les modèles ont en commun au moins un attribut de cette dimension :

Définition 3.3 : Le matching de deux profils au niveau modèle est égal à l'intersection des modèles qui ont servi à leur instanciation.
Soient P_1 et P_2 deux profils.

Match(Model(P_1), Model(P_2)) = Model(P_3) / $\forall$ a $\in$ Attributes(P_3), a $\in$ Attributes(P_1) $\wedge$ a $\in$ Attributes(P_2) $\wedge \forall$ d $\in$ Dimensions(P_3), $\exists$ $a_i \in$ Attributes(P_3) t.q. $a_i \in$ d

Le matching du modèle d'un profil utilisateur avec celui d'un système d'information est une manière simple de valider le profil utilisateur par rapport au système. En effet, en supposant que le profil du système contient tous les attributs reconnus par le système, le matching entre ce profil et ceux des utilisateurs permet de trouver les parties des profils utilisateur que le système est capable de prendre en compte.

Le résultat du matching des deux profils de l'exemple P_V et P_M au niveau modèle est égal au modèle de P_M parce que son modèle est inclus dans celui de P_V. Supposons que P_V est un profil utilisateur et P_M est le profil d'un système. Une validation de P_V par rapport à P_M en utilisant le matching détermine que les prédicats du domaine d'intérêt, la fraîcheur des données et leurs formats sont les seules informations du profil utilisateur qui peuvent être prises en compte par le système.

Le matching de modèles de profils permet de mettre en facteur les parties communes de leur structure, mais ne donne aucun renseignement sur les correspondances des valeurs de leurs attributs. Pour trouver ces correspondances on utilise le matching des instances. Cette

primitive rend une instance de profil où les valeurs de chaque attribut sont l'intersection des valeurs de cet attribut dans les deux instances initiales :

Définition 3.4 : Le matching de deux profils au niveau valeurs est égal à la restriction du matching de leurs modèles aux seuls attributs ayant deux à deux des valeurs non disjointes dans les deux profils. La valeur de chaque attribut dans le profil résultat est l'intersection des valeurs de cet attribut dans les deux profils initiaux.
Soient P_1 et P_2 deux profils.

$$\mathbf{Match(Value(P_1), Value(P_2)) = P_3 / \ \forall\ a \in Model(P_3), a \in Match(Model(P_1), Model(P_2)) \wedge Values(a, P_3) = Values(a, P_1) \cap Values(a, P_2) \neq \varnothing}$$

Le modèle du profil résultat est inclus dans le modèle qui résulte du matching des modèles des deux profils. En effet, le matching des instances ajoute une condition supplémentaire sur la présence d'un attribut dans le profil résultat. À savoir, chaque attribut doit être instancié. Rappelons que l'instanciation d'un attribut est faite en lui associer des valeurs. Dans ce cas, le modèle du profil obtenu par le matching des instances est une restriction du résultat du matching au niveau modèle aux seuls attributs dont l'intersection des valeurs n'est pas nulle.

Faire le matching entre les valeurs d'un profil utilisateur et le profil d'un système permet d'effectuer une validation simple de ce profil par rapport au système. Dans ce cas, l'intersection des valeurs des deux profils correspond aux valeurs du profil utilisateur que le système reconnaît.

Le résultat du matching des valeurs de P_V et P_M est présenté sur la Figure 3.18. C'est une instance de profil qui a le même modèle que P_M et où les valeurs des attributs sont les intersections des valeurs de P_V et P_M.

Match(Value(P_V), Value(P_M))
Domaine d'intérêt
Prédicat = {
Film.genre = 'action'
Film.genre = 'dessin animé' }
Qualité
fraîcheur < 1/2 jour
Données de livraison
format des fichiers = {
PDF, PS }

Figure 3.18 : Résultat de Matching(Value(P_V), Value(P_M))

Si on suppose que le matching représente la validation du profil P_V par rapport au système dont le profil est P_M, alors le profil de la Figure 3.18 correspond à la partie utilisable de P_V sur le système.

3.2.3 Différence de profils

La primitive de différence (MisMatch) permet de trouver le complément de l'intersection entre deux profils. À la différence de l'équivalence et du matching qui sont commutatives, la différence entre deux profils dépend du sens dans lequel elle est appliquée. La différence au niveau modèle d'un profil P_1 par rapport à un autre profil P_2 est égale à la partie de Model(P_1) qui n'apparaît pas dans Matching(Model(P_1), Model(P_2)).

Définition 3.5 : La différence d'un profil P_1 par rapport à un autre profil P_2 au niveau modèle est égale à la partie de Model(P_1) qui n'apparaît pas dans Model(P_2)

Soient P_1 et P_2 deux profils.

MisMatch(Model(P1), Model(P2)) = Model(P_3) / ∀ a ∈ Attributes(P_3), a ∈ Attributes(P_1) ∧ a ∉ Attributes(P_2) ∧ ∀ d ∈ Dimensions(P_3), ∃ a_i ∈ Attributes(P_3) t.q. a_i ∈ d

Par exemple le résultat de MisMatch(Model(P_V), Model(P_M)) est le modèle de profil de la Figure 3.19. Il contient les attributs de P_V qui n'apparaissent pas dans P_M ainsi que les dimensions auxquelles appartiennent ces attributs. De même le résultat de MisMatch(Model(P_M), Model(P_V)) est un modèle de profil vide car Model(P_M) est inclus dans Model(P_V).

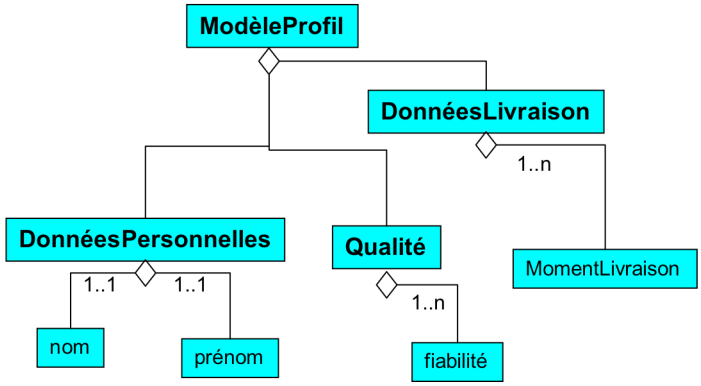


Figure 3.19 : Résultat de MisMatch(Model(P_V), Model(P_M))

Finalement, le résultat de la différence des valeurs de deux instances de profils est une instance de profil dont les attributs sont ceux du Matching(Model(P_1), Model(P_2)) qui ont des différences dans leurs valeurs et les valeurs sont égales à ces différences.

Définition 3.6 : La différence d'un profil P_1 par rapport à un autre profil P_2 au niveau valeurs est une instance de profil où :

- le niveau modèle correspond aux attributs du matching de Model(P_1) et Model(P_2) qui ont des valeurs dans P_1 différentes de celles dans P_2,
- les valeurs des attributs sont les valeurs de Value(P_1) qui n'apparaissent pas dans Value(P_2).

Soient P_1 et P_2 deux profils.

MisMatch(Value(P_1), Value(P_2)) = P_3 / ∀ a ∈ Model(P_3), a ∈ Match(Model(P_1), Model(P_2)) ∧ Values(a, P_3) = (Values(a, P_1) \ Values(a, P_1) ∩ Values(a, P_2)) ≠∅

MisMatch(Value(P_V), Value(P_M))	**MisMatch(Value(P_M), Value(P_V)**

Domaine d'intérêt
Prédicat = {
Film.producteur = 'Coppola' }
Qualité
fraîcheur (>1/2 jour ∧ < 1 jour)
∨ (>1/2 jour ∧ < 2 ans)
Données de livraison
format des fichiers = {
DOC }
	Domaine d'intérêt
	Prédicat = {
	Film.genre = 'triller' }
	Données de livraison
	format des fichiers = {
	XML, HTML }

Figure 3.20 : Résultat de la différence des valeurs des profils P_V et P_M

Le résultat de la différence entre les valeurs de P_V et P_M est représenté sur la Figure 3.20. Le premier constat qu'on peut faire est que le résultat n'est pas le même selon l'ordre dans lequel la

primitive est appliquée. Par exemple l'utilisateur souhaite recevoir des fichiers au format DOC qui n'est pas reconnu par le système et le système propose des formats que l'utilisateur n'a pas renseignés dans son profil (XML et HTML). Il faut remarquer également que dans les deux cas, les attributs des profils résultats sont inclus dans le matching des modèles des profils initiaux.

Afin de résumer le fonctionnement des primitives d'appariement de profils, le Tableau 3.2 les présente de façon informelle.

	Niveau modèle	Niveau valeurs
Équivalence	**Equivalence(Model(P_1), Model(P_2)) → booléen** Deux profils sont équivalents au niveau modèle s'il ont été obtenus par instanciation du même modèle type de profil	**Equivalence(Value(P_1), Value(P_2)) → booléen** Deux profils sont équivalents par leurs valeurs s'ils sont équivalents au niveau modèle et si tous leurs attributs possèdent deux à deux les mêmes valeurs
Match	**Match(Model(P_1), Model(P_2)) → Model(P_3)** rend un modèle de profil dont les dimensions, sous-dimensions et attributs appartiennent à la fois à Model(P_1) et à Model(P_2).	**Match(Value(P_1), Value(P_2)) → P_4** rend une instance de profil P_4 où Model(P_4) est une restriction de Model(P_3) aux seuls attributs ayant deux à deux des valeurs non disjointes dans Value(P_1) et Value(P_2). L'intersection des valeurs des attributs dans Value(P_1) et Value(P_2) représentent les valeurs de Value(P_4).
MisMatch	**MisMatch(Model(P_1), Model(P_2)) → Model(P_5)** rend un modèle de profil Model(P_5) = Model(P_1) \ Match(Model(P_1), Model(P_2))	**MisMatch(Value(P_1), Value(P_2)) → P_6** rend une instance de profil P_6 où Model(P_6) contient les attributs de Model(P_1) qui appartiennent aussi à P_3 et qui ont des valeurs disjointes dans Value(P_1) et Value(P_2). Les valeurs de Value(P_1) qui n'apparaissent pas dans Value(P_2) représentent les valeurs de Value(P_6).

Tableau 3.2 : Tableau récapitulatif des définitions des primitives d'appariement de profils

4. Plateforme de gestion de méta modèles

Les modèles génériques de profils, de contexte et de préférences ont été implémentés pour former une plateforme de gestion de méta modèles. L'objectif de la plateforme est de supporter les modèles génériques présentés dans la section 2 ainsi que l'ensemble des opérations définies sur ces modèles. Cette section présente l'architecture et les principales fonctionnalités de la plateforme.

4.1 Architecture de la plateforme

Comme nous l'avons mentionné précédemment, le travail décrit dans cette thèse s'inscrit dans le cadre du projet Accès Personnalisé à des Masses de Données (APMD). Par conséquent, la plateforme de gestion de méta modèles s'inscrit dans le contexte architectural de ce projet qui est présenté sur la Figure 3.21. L'architecture du projet APMD comporte deux parties principales : une dédiée à la gestion et la manipulation de profils et de contextes (ce qui inclut les préférences exprimées sur ces profils et contextes) et une autre pour l'acquisition et l'évaluation de la qualité. Un accès adaptatif à l'information s'effectue en utilisant les informations des deux parties ainsi que les requêtes des utilisateurs. Une fois les

résultats personnalisés retournés à l'utilisateur, son comportement est observée pour mettre à jour son profil (Feedback).

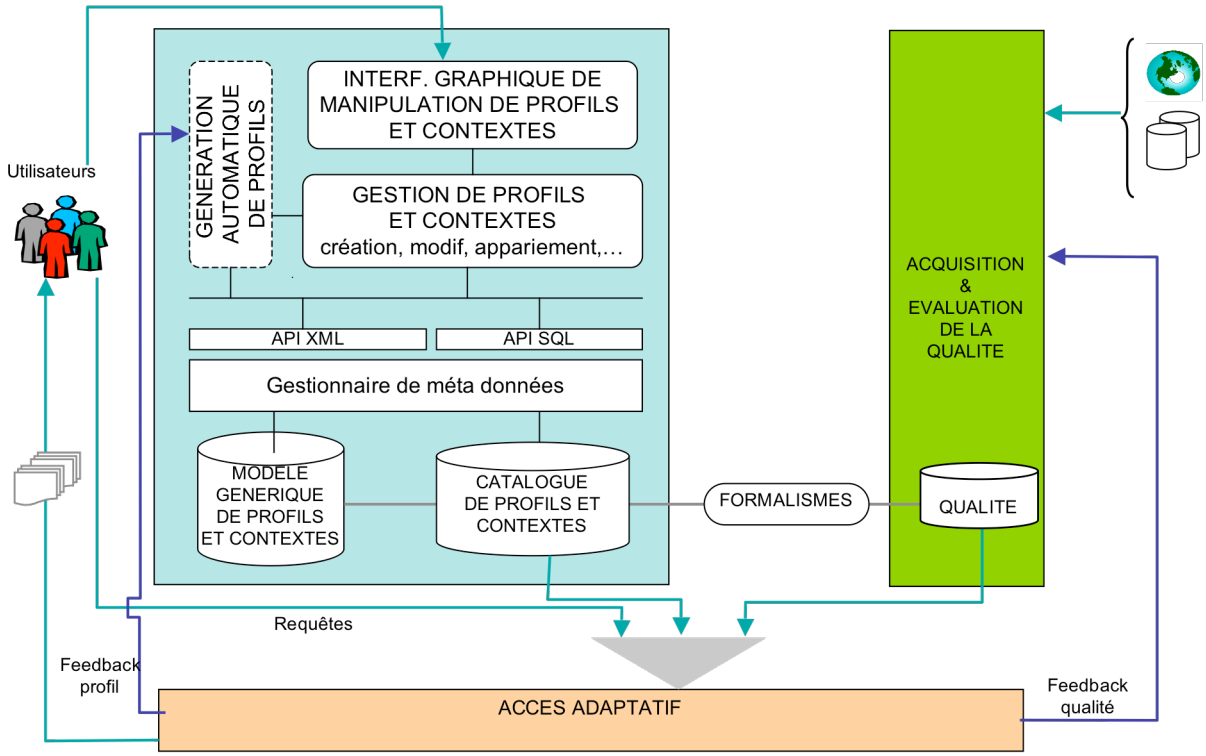


Figure 3.21 : Architecture du projet APMD

Le développement que nous avons réalisé se situe au niveau de la partie gauche de l'architecture du projet APMD qui concerne la gestion des profils et des contextes. Une interface graphique permet de manipuler les profils, les contextes et les préférences. Le modèle générique de profils et de contexte ainsi que leurs instances sont implémentés dans une base de données Oracle. La connexion entre la base de données et son environnement applicatif est faite, au choix, par une API SQL pour une vision relationnelle de la base ou une API XQuery pour une vision XML de la base.

4.2 Fonctionnalités de la plateforme

Un important effort d'implémentation est fait pour réaliser les gestionnaires de profils et de contextes. Le gestionnaire des préférences est en cours de développement. Actuellement, la plateforme supporte l'expression de deux types de préférences simples : le poids et la comparaison binaire de deux éléments (>>). L'ajout d'une préférence sur un élément d'un profil ou d'un contexte se fait par un simple click sur l'élément ce qui déclanche l'affichage d'un menu contextuel dans lequel on peut choisir d'ajouter une préférence.

4.2.1 Fonctionnalités du gestionnaire de profils

Le gestionnaire de profils implémente l'ensemble des opérations définies dans la section 3. Une interface graphique permet de créer et de manipuler manuellement les profils. La fenêtre principale de cette interface permet de gérer à la fois les profils utilisateurs, ceux des systèmes d'information et des sources de données (Figure 3.22). Horizontalement, la fenêtre est divisée en trois parties chacune permettant de gérer une catégorie de profils (utilisateur, système ou sources). Ensuite, la fenêtre est divisée verticalement en deux parties : (i) la partie gauche qui affiche les modèles génériques des profils et (ii) la partie droite réservée aux instances de profils. Un modèle générique regroupe l'ensemble des dimensions, des sous

dimensions et des attributs qu'un modèle de profil peut contenir. Par exemple, les dimensions du profil générique de l'utilisateur correspondent aux 5 dimensions présentées dans la section 2.1 (Domaine d'Intérêt, Données personnelles, Données de Qualité, Données de Livraison et Données de Sécurité). Pour garantir la flexibilité du modèle générique, l'interface graphique permet de définir de nouvelles dimensions, sous dimensions et de nouveaux types d'attributs.

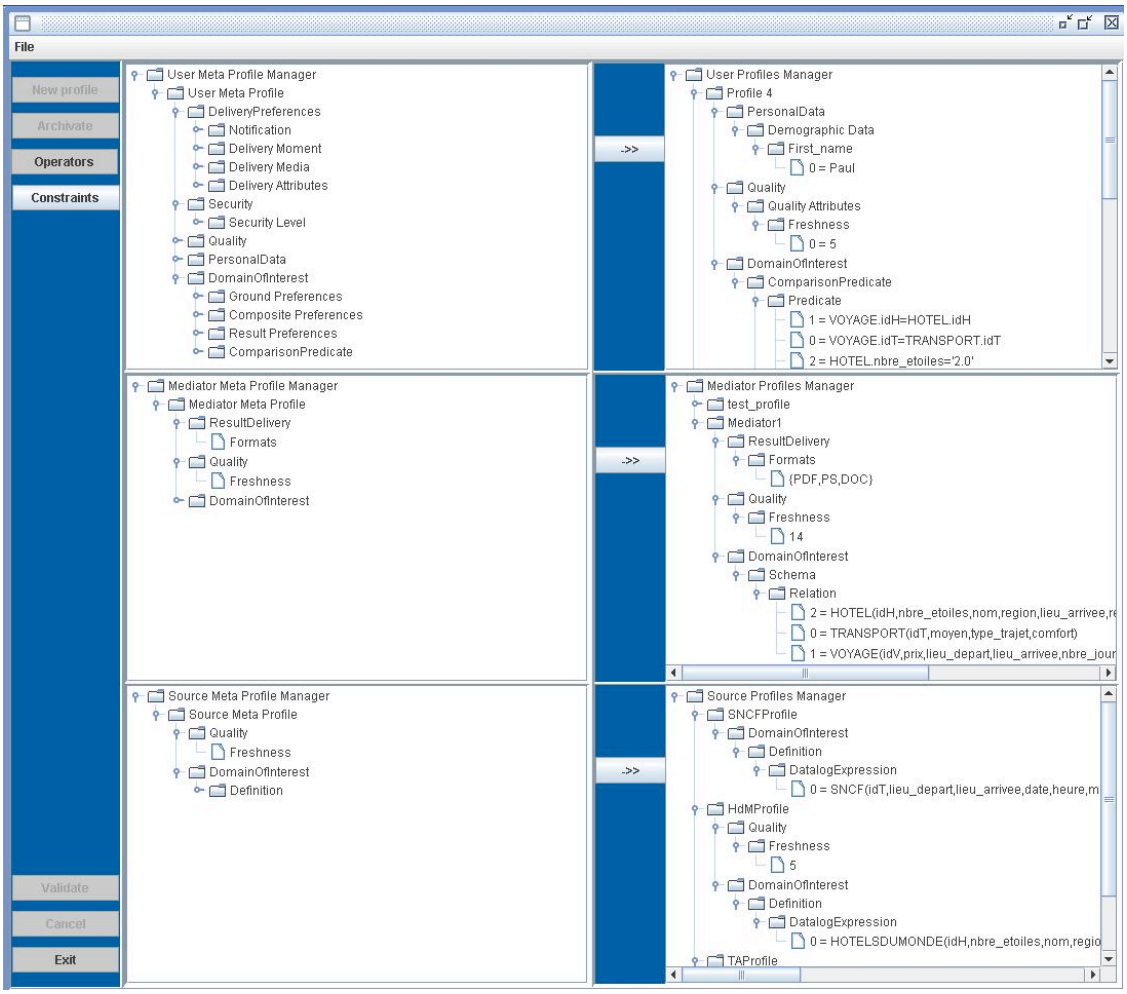


Figure 3.22 : Fenêtre principale du gestionnaire de profils

Les principales fonctionnalités du gestionnaire de profils sont :

- l'instanciation de profils,
- mise à jour des modèles et des instances,
- archivage des instances,
- l'appariement de profils. Les trois primitives des opérations d'appariement sont implémentées. Elles s'appliquent aussi bien sur les modèles que sur les valeurs.

Instanciation de profils

L'instanciation d'un profil est faite en deux temps : (i) choix de la structure du profil (modèle type) et (ii) attribution de valeurs aux attributs du profil (instanciation de modèle).

La première étape (sélection de la structure d'un profil) peut être faite de trois manières différentes :

- recopie de l'ensemble du modèle générique,

- choix de la même structure qu'une instance de profil déjà existante (modèle type existant) ou
- sélection d'une structure personnalisée à partir du modèle générique (création d'un nouveau modèle type).

La recopie de la structure d'un profil existant (profil générique ou instance de profil) est faite automatiquement sans prendre les valeurs éventuelles de ses attributs. Le choix personnalisé est fait en glissant les attributs, les sous dimensions ou les dimensions voulus du profil générique vers le nouveau profil. La recopie d'une sous dimension entraîne celle de l'ensemble de ses attributs, de même que le choix d'une dimension équivaut au choix de l'ensemble de ses sous dimensions et attributs.

La deuxième étape de la création d'un profil revient à attribuer des valeurs aux attributs. Ceci peut être fait par le module de génération automatique, par apprentissage ou datamining par exemple, ou manuellement par l'utilisateur via l'interface graphique. La construction automatique d'un profil est un processus complexe qui dépasse le cadre de cette thèse et ne fera pas l'objet d'une discussion.

Mise à jour des profils

Le gestionnaire de profils offre des fonctionnalités de mise à jour classiques qui sont : l'insertion, la suppression et la modification de la structure et des valeurs des profils. La seule règle à respecter est que le modèle de profil générique doit inclure l'union des structures de toutes les instances de profils. Pour maintenir le profil générique le plus complet possible, chaque insertion d'une nouvelle dimension, sous dimension ou attribut dans une instance de profil se fait par le modèle générique. Par exemple si un utilisateur veut ajouter un attribut « *revenu* » à ces données personnelles, il est obligé de créer cet attribut dans le modèle générique de profil utilisateur pour ensuite le recopier dans son profil. De cette manière la typologie des données est préservée (par exemple le nom d'un utilisateur est une chaîne de caractères dans l'ensemble des instances de profils utilisateurs).

Archivage de profils

L'archivage de profils permet de sauvegarder le contenu d'une instance de profil. Lors de cette sauvegarde, la date de création de l'archive est sauvegardée avec l'ensemble des valeurs du profil. Plusieurs archives peuvent être créées pour le même profil ce qui permet de suivre l'évolution des préférences d'un utilisateur ou d'analyser son comportement. Les archives peuvent être visualisées et il est possible de remplacer le contenu actuel du profil par une de ses archives. Ceci peut être très utile si on veut invalider une mise à jour non pertinente du profil utilisateur et revenir à une version précédente.

4.2.2 Fonctionnalités du gestionnaire de contextes

La fenêtre du gestionnaire de contexte comporte deux parties principales. La partie supérieure est réservée à l'instanciation de contextes tandis que la partie inférieure permet de gérer les relations entre les contextes et les profils utilisateur (Figure 3.23). Les principales fonctionnalités de la plateforme de gestion de contextes sont :

- l'instanciation de contextes,
- la mise à jour des modèles et des instances de contextes,
- la contextualisation des valeurs et des préférences des profils utilisateurs,
- l'adaptation d'un profil utilisateur par rapport à un contexte particulier.

Instanciation de contextes

L'instanciation de contextes suit le même principe que celle des profils. D'abord un modèle type de contexte est choisi et ensuite des valeurs sont attribuées aux attributs de ce modèle. La seule différence entre les deux gestionnaires est que celui des contextes fait apparaître de façon explicite le niveau modèle. Dans le gestionnaire de profils, ce niveau est intégré dans le niveau instance. Pour réutiliser un modèle type de profil, on recopie le modèle d'une de ses instances. Dans le gestionnaire de contextes, chaque instance est créée à partir d'un modèle type prédéfini et visible. Ce choix d'implémentation est motivé par le fait qu'en général chaque utilisateur créer son profil en fonction de ses propres besoins et il est rare que deux utilisateurs distinct aient exactement les mêmes besoins. Souvent c'est l'utilisateur lui-même qui réutilise le modèle d'un de ses profils. Il est donc plus naturel pour un utilisateur de recopier le modèle d'une de ses instances. En ce qui concerne les contextes, leur contenu est souvent fixé par l'application et les utilisateurs se voient « imposer » un modèle de contexte à instancier. Dans ce cas, il est indispensable de faire apparaître les modèles types afin de proposer aux utilisateurs un catalogue de contextes.

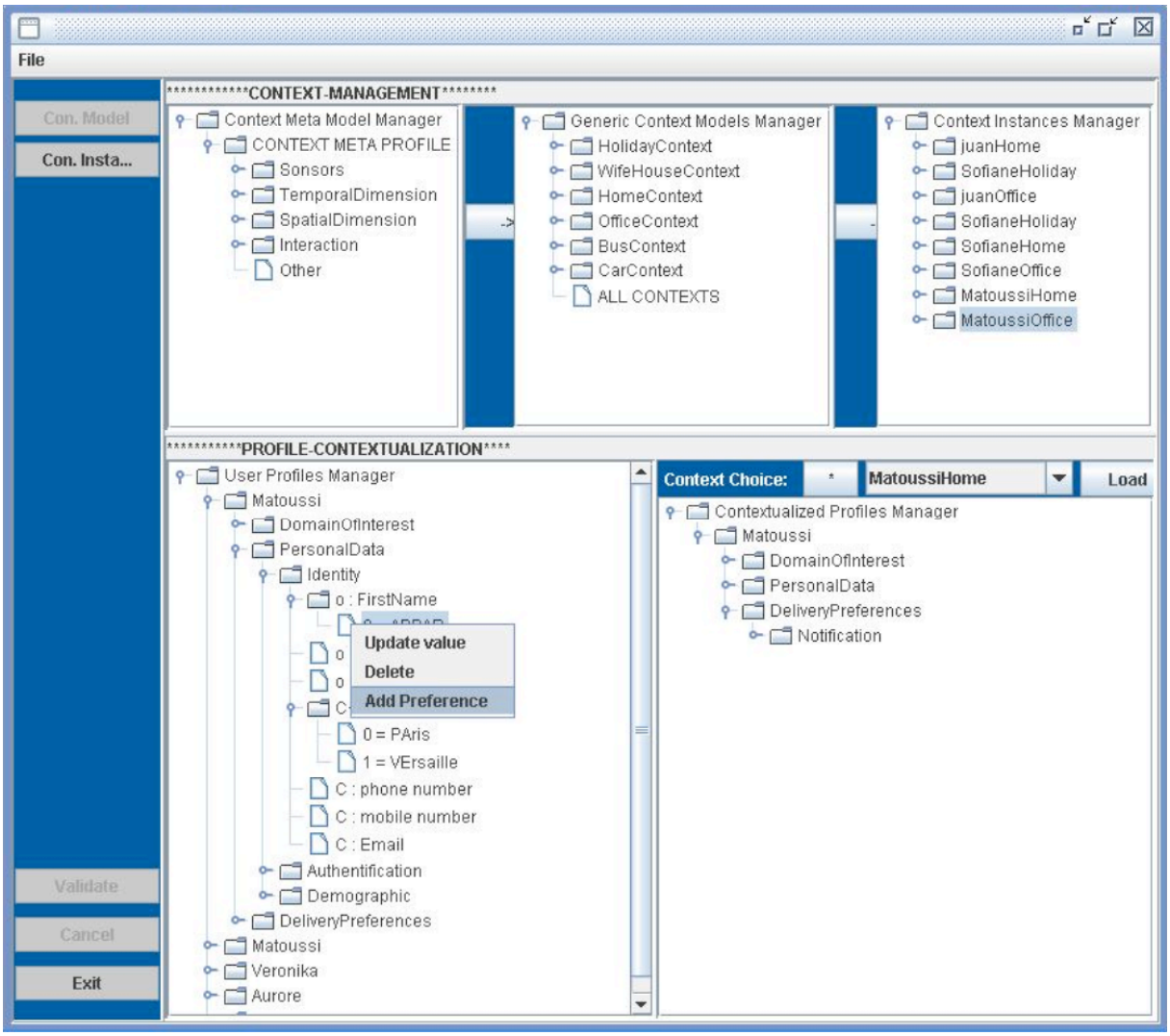


Figure 3.23 : Fenêtre du gestionnaire de contextes

Mise à jour de contextes

Le gestionnaire de contextes offre les mêmes fonctionnalités de mise à jour que le gestionnaire de profils. Il est donc possible d'insérer, de supprimer ou de modifier un élément du contexte. Comme pour le modèle générique de profils, le modèle générique de contexte

doit rester le plus complet possible et l'insertion d'un nouvel attribut se fait toujours à travers de ce modèle (partie supérieure gauche de la fenêtre).

Contextualisation

La première fonctionnalité spécifique au gestionnaire de contexte est la contextualisation. Elle revient à rendre dépendant du contexte un attribut ou une préférence d'un profil utilisateur. Ceci est fait dans la partie inférieure gauche de la fenêtre de gestion des contextes (Figure 3.23). Cette zone affiche les instances des profils utilisateurs. L'ajout d'une valeur ou d'une préférence à un profil utilisateur se fait en choisissant l'instance de contexte de cet utilisateur dans laquelle la valeur ou la préférence sera appliquée.

Adaptation d'un profil par rapport à un contexte

Finalement, la partie inférieure droite de la fenêtre sert à afficher le résultat de l'adaptation des profils utilisateurs. Ceci est fait en choisissant une instance de profil utilisateur à partir de la zone gauche de la fenêtre. Une fois l'instance choisie, elle s'affiche dans la zone droite et en même temps l'ensemble des contextes de l'utilisateur auquel appartient le profil s'affichent dans la « ComboBox » qui se trouve en haut de cette zone. En choisissant un des contextes disponibles, l'instance du profil utilisateur est automatiquement adaptée. Cette opération revient à afficher les attributs et les valeurs non contextualisés du profil ainsi que les valeurs des attributs contextualisés qui s'appliquent au contexte choisi.

Les gestionnaires de profils et de contextes présentés dans cette section permettent de les manipuler manuellement. Cependant un des objectifs de la personnalisation est de faire intervenir l'utilisateur le moins souvent possible. C'est la raison pour laquelle les gestionnaires ont été conçus comme des APIs qui peuvent être utilisées par un autre programme via les opérateurs et les fonctions définis dans ces APIs.

5. Conclusion

L'élaboration de modèles génériques qui permettent de représenter les connaissances qui décrivent l'utilisateur, l'environnement dans lequel il se trouve et ses préférences est le premier pas vers la construction de systèmes de personnalisation capables de délivrer des informations selon les préférences d'un utilisateur. Dans ce chapitre nous avons défini des modèles génériques de profil, de contexte et de préférences qui permet de structurer les informations nécessaires pour la description des besoins et des préférences d'un utilisateur. Ces modèles regroupent un grand nombre de paramètres nécessaires à divers scénarios de personnalisation. Ils sont complétés par un ensemble d'opérations de gestion de profils qui serviront à une manipulation directe ou via des APIs. Nous avons également présenté un prototype de gestion de profils et de contextes qui implémente les modèles génériques et les opérateurs associés. Cette plateforme peut servir à créer un catalogue de profils et de contextes qui sera utilisé pour tester différents scénarios de personnalisation.

Chapitre 4. Reformulation de requêtes

Ce chapitre décrit et évalue deux approches de reformulation de requêtes dans un contexte de médiation en tenant compte des préférences de l'utilisateur et des descriptions des sources de données. Il propose une présentation des algorithmes de réécriture et d'enrichissement sur lesquels sont basées les deux approches. Les avantages et les risques que représentent les approches sont discutés grâce à deux métriques d'évaluation définies dans ce chapitre.

1. Introduction et contexte

Le travail présenté dans ce chapitre s'inscrit dans un contexte de médiation. Un système de médiation est un système d'intégration de données, qui offre un accès transparent à des sources de données distribuées et hétérogènes. Il est généralement défini par quatre composants: un schéma virtuel, un ensemble de liens sémantiques reliant ce schéma aux sources de données (appelés aussi requêtes de médiation), un module de réécriture de requêtes utilisateur et un module d'intégration de données qui réalise les opérations multi sources (jointures, union, agrégat) à partir des résultats partiels calculés par les systèmes sources.

L'introduction de la personnalisation de l'accès dans une telle architecture nécessite son enrichissement. Chaque utilisateur peut être décrit par un ou plusieurs profils définissant ses centres d'intérêts. L'évaluation d'une requête utilisateur doit être faite relativement à ce profil. La requête ne traduit alors plus qu'une expression approchée du besoin de l'utilisateur. Elle est enrichie par le profil qui réduit la taille des résultats produits.

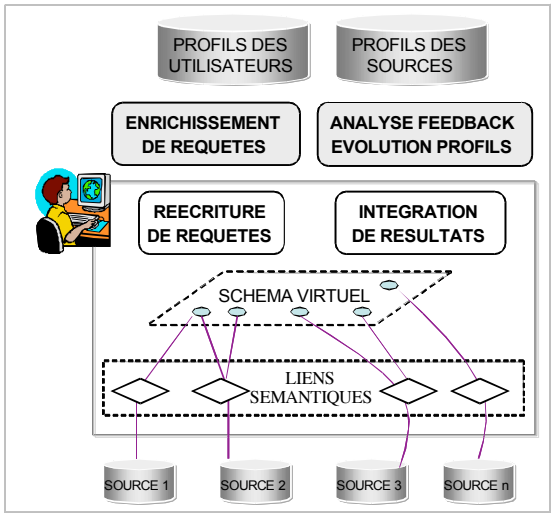


Figure 4.1 : Architecture d'un système de médiation personnalisé

La Figure 4.1 décrit l'architecture d'un médiateur personnalisable. Les composants colorés en gris sont les compléments aux composants classiques d'une architecture de médiation représentés par les boîtes blanches. La base de profils des sources décrit les méta données caractérisant chaque source de données (structure de données, contraintes d'intégrité, facteurs de qualité, événements d'évolution, ...). Le profil d'un utilisateur est décrit par le modèle présenté dans le chapitre 3. Il inclut le centre d'intérêt de l'utilisateur, le contexte

d'émission de la requête, le niveau de qualité désiré, l'historique des interactions ainsi que diverses préférences sur ces dimensions.

Dans ce contexte architectural, l'évaluation d'une requête utilisateur se fait selon le cycle de vie de la Figure 4.2. Chaque requête utilisateur est reformulée en exploitant, d'une part, le profil de l'utilisateur et, d'autre part, les profils des sources de données. Les sous-requêtes obtenues sont exécutées sur les sources et leurs résultats intégrés au niveau du médiateur. La personnalisation intervient à chacune des étapes de ce cycle de vie, y compris dans la présentation des résultats.

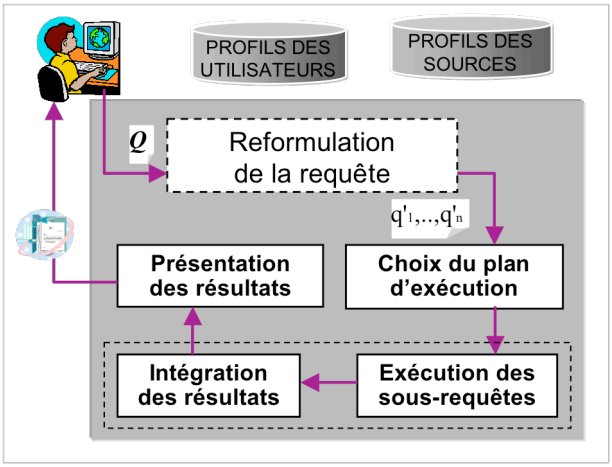


Figure 4.2 : Cycle de vie d'une requête personnalisée

La contribution du chapitre concerne la reformulation de requêtes. La principale question à laquelle nous allons répondre est comment reformuler la requête de l'utilisateur en tenant compte à la fois de ses préférences et des sources de données. Il existe des techniques permettant d'adresser chacun de ces problèmes séparément. L'enrichissement permet d'intégrer les préférences de l'utilisateur dans sa requête et la réécriture permet de traduire la requête en un ensemble d'expressions pouvant être évaluées sur les sources de données. Mais aucune de ces techniques ne permet à elle seule de répondre à la question posée. Par conséquent, l'accès personnalisé à des sources de données multiples nécessite une étape de reformulation qui combine les deux techniques pour produire des requêtes exécutables sur les sources de données et qui intègre le profil utilisateur.

Dans notre contexte, l'enrichissement et la réécriture ne sont pas indépendants. En effet, ces deux algorithmes peuvent ajouter des restrictions à la requête (préférences du profil pour l'enrichissement et restrictions venant des définitions des sources de données pour la réécriture). Or, comme leur comportement dépend des prédicats de la requête qu'ils reformulent, le résultat final dépend de l'ordre dans lequel les deux algorithmes sont appliqués.

La remarque précédente nous conduit à proposer et à analyser deux approches pour la reformulation d'une requête : la première où la réécriture s'applique sur le résultat de l'enrichissement, la seconde où l'enrichissement s'applique sur le résultat de la réécriture.

La problématique étudiée dans ce chapitre étant l'étude et l'évaluation de la composition des algorithmes de réécriture et d'enrichissement, nous faisons deux hypothèses : (i) parmi les métadonnées décrivant les sources, seules leurs définitions sont utilisées et, (ii) les problèmes liés à l'hétérogénéité sont supposés résolus, i.e. les noms de relations ou d'attributs identiques correspondent à des concepts identiques et les noms différents correspondent à des concepts différents.

Ce chapitre est organisé de la manière suivante. La section 2 rappelle les principes des algorithmes de reformulation de requêtes (réécriture et enrichissement) que nous utilisons. La section 3 décrit les deux approches de médiation personnalisée. La section 4 définit deux métriques qui sont la couverture et l'utilité, et compare ces approches en se basant sur les résultats que nous avons obtenus sur un échantillon de requêtes. La section 5 conclut le chapitre en discutant de la généralisation de l'approche.

2. Techniques de reformulation de requêtes

Cette section rappelle les mécanismes de réécriture et d'enrichissement de requêtes que nous avons retenus pour illustrer l'importance de la personnalisation dans un système d'intégration de données. Les deux algorithmes traitent des requêtes conjonctives exprimées sur un schéma relationnel.

2.1 Enrichissement

L'enrichissement d'une requête exploite 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. La méthode la plus récente et la plus aboutie est celle de [KI04a, KI05b] qui est résumée dans cette section.

Dans l'approche de Koutrika et Ioannidis [KI04a], 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. Il est spécifié par un nombre réel compris entre 0 et 1. Prenons par exemple une base de données dont le schéma est le suivant :

Exemple 4.1 : Exemple de schéma d'une base de données (S_v)

```
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)
```

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 4.2 : 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 (h) 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 G = (V, E) où V est l'ensemble de nœuds et E est l'ensemble d'arcs. Dans ce graphe, Il y a trois types de nœuds :

- nœuds de relations : un pour chaque relation du schéma,
- nœuds d'attributs : un pour chaque attribut de chaque relation du schéma,
- nœuds de valeurs : un pour chaque valeur des prédicats du profil utilisateur.

De leur côté les arcs peuvent être des arcs de sélection entre un nœud d'attribut et un nœud de valeur et des arcs de jointure entre deux noeuds d'attribut. Les arcs possèdent un poids qui correspond à l'intérêt du prédicat représenté par l'arc. La Figure 4.3 montre la représentation graphique du profil P_1.

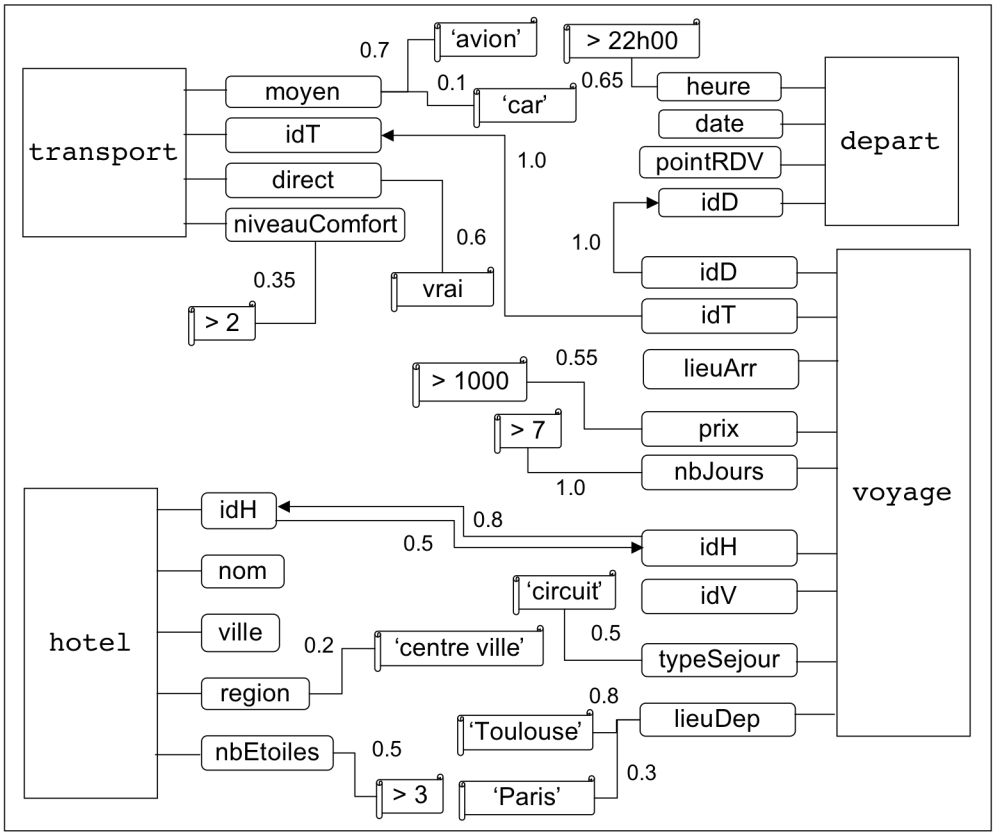


Figure 4.3: Représentation graphique du profil utilisateur

Cette représentation du profil utilisateur a été étendue en ajoutant un poids supplémentaire aux prédicats de sélection [KI05b]. Dans ce modèle, le degré d'intérêt d'un prédicat de sélection *q* sur un attribut *A* est défini pour chaque valeur $u \in D_A$ (domaine de définition de A), par un couple de fonctions $d_T(u)$ et $d_F(u)$ telles que :

doi(q) = ($d_T(u)$, $d_F(u)$) où $d_T(u)$, $d_F(u) \in [-1, 1]$ et $d_T(u)*d_F(u) \leq 0$.

Dans cette représentation, $d_T(u)$ exprime l'intérêt de l'utilisateur pour la présence de la valeur *u* tandis que $d_F(u)$ montre son intérêt pour l'absence de cette valeur. Par exemple si l'utilisateur vaut absolument loger dans un hôtel au centre ville, il peut le faire en donnant une grande valeur pour la présence du prédicat (par exemple d_T(HOTEL.region = 'centre ville') = 0.8) et une petite valeur négative pour son absence (par exemple d_F(HOTEL.region = 'centre ville') = -0,7).

Les signes des degrés d'intérêt $d_T(u)$ et $d_F(u)$ expriment le « signe » de la préférence. Lorsque $d_T(u) > 0$, d_T montre à quel point l'utilisateur est intéressé par la présence de u, alors que si $d_T(u) < 0$, d_T désigne le degré d'antipathie de l'utilisateur pour u. De même une valeur positive pour $d_F(u)$ montre l'intérêt de l'utilisateur pour l'absence de la valeur u, alors qu'une valeur négative pour $d_F(u)$ exprime une antipathie pour l'absence de cette valeur. Finalement, la valeur 0 exprime une indifférence pour la présence (d_T) ou l'absence (d_F) d'une valeur.

Les fonctions d_T et d_F peuvent exprimer des préférences exactes ou élastiques en fonction de la nature des valeurs contenues dans le domaine de définition D_A de l'attribut sur lequel elles sont exprimées. Si D_A contient des valeurs catégoriques (discrètes) qui sont toutes indépendantes les unes des autres, alors la préférence est exacte i.e. elle peut être complètement satisfaite ou pas de tout. Par contre les préférences exprimées sur des valeurs numériques peuvent être continues et de ce fait peuvent être satisfaites partiellement. L'élasticité est capturée par la forme des fonctions $d_T(u)$ et $d_F(u)$. Des fonctions constantes

sont utilisées pour les préférences exactes, tandis que pour les préférences élastiques, plusieurs fonctions possibles peuvent être utilisées de sorte à attribuer un degrés d'utilité en fonction de la valeur de l'attribut A. La forme des fonctions utilisées dans [KI05b] est donnée sur la Figure 4.4. Elles sont notées $e_{(d)}$ où *d* désigne la plus grande valeur d'intérêt que la fonction peut attribuer à une valeur de D_A.

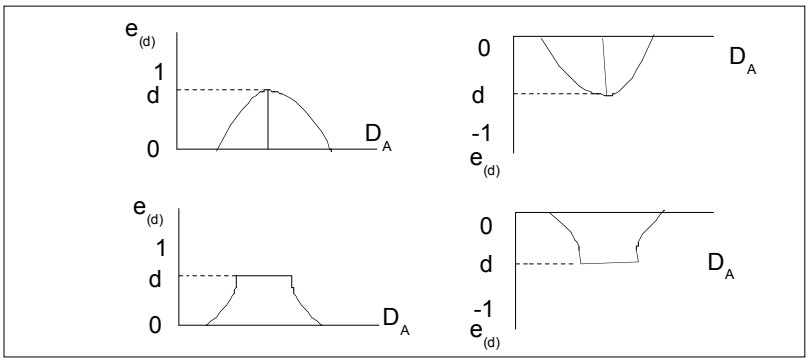


Figure 4.4 : Exemples de fonctions élastiques

Si on étend le profil P_1 de l'Exemple 4.2, on peut obtenir le profil suivant :

Exemple 4.3 : Exemple de profil étendu

`Profil P2 : {`	`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, -0.4)`	`(e)`
	`VOYAGE.lieuDep = 'Toulouse'`	`(0.8, -0.5)`	`(f)`
	`TRANSPORT.moyen='avion'`	`(0.7, -0.1)`	`(g)`
	`DEPART.heure > 22h00`	`(0.65, -0.9)`	`(h)`
	`TRANSPORT.direct=VRAI`	`(0.6, 0)`	`(i)`
	`VOYAGE.prix < 1000`	`(0.55, -1.0)`	`(j)`
	`HOTEL.nbEtoiles > 3`	`(0.5, -0.5)`	`(k)`
	`VOYAGE.typeSejour = 'circuit'`	`(0.5, -1.0)`	`(l)`
	`TRANSPORT.niveauConfort > 2`	`(0.35, e-0.9)`	`(m)`
	`VOYAGE.lieuDep = 'Paris'`	`(0.3, 0)`	`(n)`
	`HOTEL.region = 'banlieue'`	`(-0.2, 0.7)`	`(o)`
	`TRANSPORT.moyen = 'car'`	`(0.1, 0)`	`(p)`

Sur cet exemple, on voit apparaître les degrés d'intérêt lorsque les prédicats ne sont pas satisfaits. Par exemple, l'utilisateur a une forte préférence pour les voyages après 22h00 (d_T(`DEPART.heure > 22h00`) = 0.65) et veut absolument que ce prédicat soit satisfait (d_F(`DEPART.heure > 22h00`) = -0.9. De même, il a une antipathie des hôtels en banlieue (d_T(`HOTEL.region = 'banlieue'`)=-0.2), mais a une forte préférence pour les hôtels qui ne s'y trouvent pas (d_F(`HOTEL.region = 'banlieue'`)=0.7). Finalement, l'insatisfaction de l'utilisateur lorsque le niveau de confort ne dépasse pas 2 dépend du niveau de confort réel, mais ne peut pas dépasser -0.9 (d_F(`TRANSPORT.niveauConfort > 2`)= $e_{-0.9}$).

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.

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 [KI04a].

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 4.2) et une requête initiale Q_1 qui recherche des voyages de 3 jours à destination de Madrid dont la date de départ et le 10 novembre 2007 (Exemple 4.4).

Exemple 4.4 : 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 4.5 montre un exemple de tel chemin pour la requête Q_1 de l'Exemple 4.4 et le profil utilisateur P_1 de l'Exemple 4.2. Sur cette figure, les nœuds représentant des éléments de la requête initiale ont un motif hachuré et les nœuds du chemin un motif en losanges. 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.

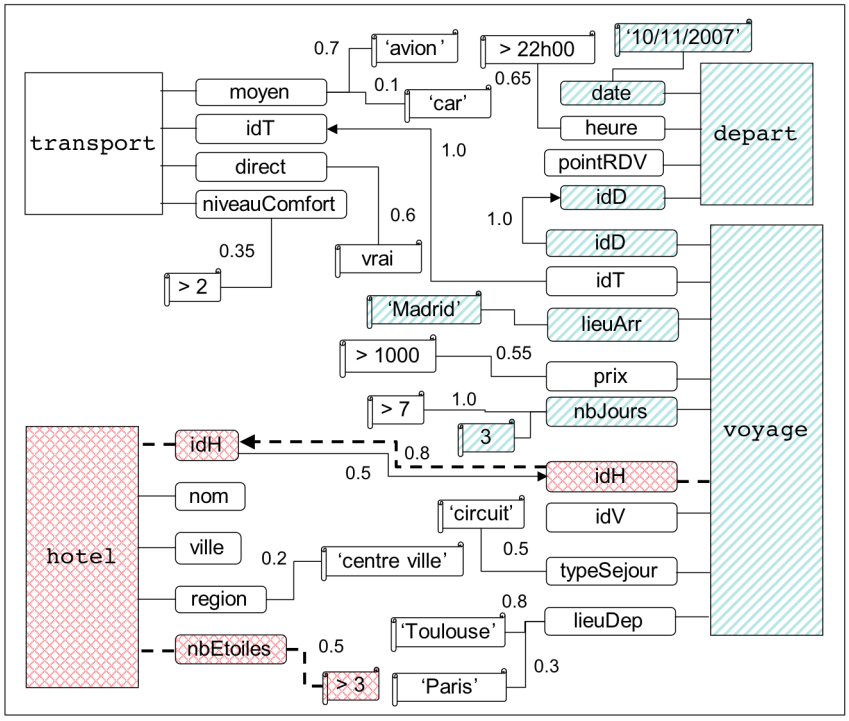


Figure 4.5 : Exemple de prédicat du profil utilisateur P_1 lié à la requête Q_1

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 [KI04a] est la multiplication des degrés d'intérêt. En

utilisant cette formule, l'intérêt de la préférences 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 4.5).

Exemple 4.5 : 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)

La dernière étape de l'algorithme d'enrichissement est l'intégration des prédicats du profil à la requête. Cette étape est guidée par les trois paramètres suivants: *K* qui est le nombre de prédicats du profil devant être pris en compte; *M* qui est le nombre de prédicats parmi les *K* qui doivent *obligatoirement* être satisfaits (ça correspond aux *M* prédicats de plus grand poids parmi les top *K*) et *L* qui est le nombre minimal de prédicats parmi les *K-M* restants que chaque tuple du résultat doit satisfaire.

Le choix des Top K préférences peut être fait selon différents critères comme par exemple : choisir au plus K prédicats, choisir les prédicats dont le poids satisfait un seuil donné, etc. [KI04a]. Le degré d'intérêt n'est pas le seul critère de sélection des prédicats du profil utilisateur. Lorsqu'on ajoute des prédicats à la requête, on augmente la pertinence des résultats produits par celle-ci, mais dans certains cas le résultat peut être vide ou le temps de réponse peut augmenter considérablement à cause du grand nombre de prédicats ajoutés [KI05a]. Les auteurs de [KI05a] proposent une approche d'optimisation sous contraintes. Dans cette approche, chaque requête Q est caractérisée par trois paramètres : (i) son coût d'exécution, (ii) le nombre de résultats qu'elle produit et (iii) le degré d'intérêt des résultats qu'elle produit. L'optimisation sous contrainte dans ce cas revient à maximiser un de ces paramètres tout en satisfaisant des contraintes sur les deux autres. Un exemple de tel problème est : maximiser le degré d'intérêt des prédicats du profil choisis tout en ayant au moins 10 tuples résultats et sans dépasser 1 seconde de calcul.

Une fois les Top K prédicats choisis, les M prédicats obligatoires (ayant le plus grand poids) sont ajoutés comme une conjonction à la requête. Pour que chaque résultat satisfasse au moins L prédicats parmi les K-M restants, il y a deux possibilités : (i) faire une seule requête contenant la disjonction de toutes les combinaisons de L prédicats parmi les (K-M) prédicats restants ou (ii) construire l'union de K-M requêtes, chacune contenant un prédicat non obligatoire, et retourner les tuples qui sont renvoyés par au moins L requêtes [KI04a]. Le nombre de requêtes qui renvoient un même tuple est calculé en utilisant une clause HAVING précédée par un GROUP BY sur les attributs sélectionnés.

Considérons la requête Q_1 de l'Exemple 4.4 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 4.5). 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 4.6 : 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 4.7 et Exemple 4.8 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 4.7 : 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 4.8 : 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 ;
```

Nous avons présenté dans cette section un rappel des techniques d'enrichissement de requêtes. Ces techniques sont utilisées dans deux approches de reformulation de requêtes que nous proposons dans ce chapitre.

Dans la suite de ce chapitre nous considérons que le profil utilisateur est composé de

prédicats ayant des poids simples (comme le profil P_1 de l'Exemple 4.2) et que le résultat du processus d'enrichissement est une seule requête (comme celle de l'Exemple 4.7).

2.2 La réécriture de requêtes

Le problème de réécriture de requêtes a été principalement abordé dans les applications d'optimisation de requêtes et d'intégration de données. La principale question à laquelle la réécriture essaie de répondre est : Etant donnée une requête Q et un ensemble de vues $\mathcal{V} = \{V_1, \ldots V_n\}$, est-il possible de répondre à Q en utilisant uniquement les données des vues $\mathcal{V}$?

Dans cette section nous nous focaliserons sur l'intégration des données dans un système distribué où les vues expriment les mappings (requêtes de médiation) entre le schéma virtuel et les sources de données. Dans un tel contexte, la réécriture de requêtes dépend de la manière de définir les mappings. Il existe deux approches principales :

- Global As View (GAV) où chaque relation virtuelle est définie par une requête sur les schémas sources et
- Local As View (LAV) où chaque relation sources est définie par une requête sur le schéma virtuel.

La première approche favorise les changements dans le schéma global. Dans ce cas, modifier une relation virtuelle revient à modifier une requête de médiation. Son inconvénient majeur vient du fait que cette approche gère mal les changements fréquents dans les sources de données ; à chaque changement de la définition d'une source il faut redéfinir tous les mappings. Contrairement à l'approche GAV, l'approche LAV facilite la prise en compte de la dynamicité des sources. En effet, modifier une source revient à modifier une seule requête. Par contre cette approche est à éviter lorsque le schéma global change fréquemment.

Nous considérons que l'approche LAV est plus réaliste parce que les systèmes de médiation interrogent souvent un ensemble de sources de données hétérogènes et autonomes. C'est la raison pour laquelle, notre travail s'inscrit dans un contexte Local As View.

La réécriture d'une requête consiste à déterminer les sources contributives pour l'exécution de la requête utilisateur et à utiliser leurs définitions pour reformuler cette requête [LRO96]. Un état de l'art sur ce sujet peut être trouvé dans [Hal01] et [CLL01]. Cette section donne quelques définitions utiles à la compréhension du problème de la réécriture de requêtes et introduit la notation Datalog qui sert à l'expression des différentes requêtes (requête initiale et requêtes de médiation). Le fonctionnement de trois algorithmes de réécriture qui sont l'algorithme de règles inversées, l'algorithme Bucket et l'algorithme MiniCon est ensuite présenté. Ce dernier est utilisé dans notre étude.

2.2.1 Notation et définitions

La réécriture de requêtes est basée sur les notions d'équivalence et inclusion de requêtes. Cette section présente les définitions utilisées dans [Hal01].

Définition 4.1 : Une requête Q_i est *incluse* dans une requête Q_j noté $Q_i \subseteq Q_j$, si les résultats de Q_i sont un sous-ensemble des résultats de Q_j quelle que soit la base de donnée sur laquelle ces requêtes sont évaluées.

Par exemple Q_i = SELECT * FROM VOYAGE WHERE prix < 1000 est incluse dans Q_j = SELECT * FROM VOYAGE WHERE prix < 1500 parce que Q_i est plus restrictive que Q_j.

Définition 4.2: Deux requêtes Q_i et Q_j sont *équivalentes* si $Q_i \subseteq Q_j$ et $Q_j \subseteq Q_i$.

À partir de ces définitions, la notion de récriture équivalente est définie comme suit :

Définition 4.3 : Une réécriture Q' est une réécriture équivalente d'une requête Q en utilisant un ensemble de vues $\mathcal{V} = \{V_1, \ldots, V_n\}$ si :

- Q' est exprimée uniquement sur des vues de $\mathcal{V}$ et
- Q' est équivalente à Q.

Dans le cadre des systèmes d'intégration de données, où les sources sont autonomes et peuvent devenir indisponibles, on considère plutôt des réécritures maximalement-incluses.

Définition 4.4 : Une réécriture Q' est une réécriture maximalement-incluse d'une requête Q en utilisant un ensemble de vues $\mathcal{V} = \{V_1, \ldots, V_n\}$ par rapport à un langage de requêtage $\mathcal{L}$ si :

- Q' $\in \mathcal{L}$ et est exprimée uniquement sur des vues de $\mathcal{V}$,
- Q' $\subseteq$ Q et
- il n'existe pas $Q_1 \in \mathcal{L}$, réécriture de Q t.q. Q' $\subseteq Q_1 \subseteq$ Q et Q_1 n'est pas équivalente à Q'.

À la différence des réécritures équivalentes, les réécritures maximalement-incluses dépendent de l'expressivité du langage de requêtage utilisé. Une réécriture peut être maximalement incluse dans une requête par rapport à un langage $\mathcal{L}_1$ et n'être qu'incluse par rapport à un autre langage plus expressif $\mathcal{L}_2$.

Les algorithmes de réécriture traitent des requêtes conjonctives. Dans la suite une requête conjonctive sera représentée par la notation Datalog suivante :

$$q(\overline{X}) :- r_1(\overline{X}_1), \ldots, r_n(\overline{X}_n).$$

Dans cette notation, $q, r_1, \ldots, r_n$ sont des noms de prédicats qui représentent des relations, et les $\overline{X}, \overline{X}_1, \ldots, \overline{X}_n$ sont des tuples de variables et de constantes. L'atome $q(\overline{X})$ est *l'entête* de la requête et représente sa réponse tandis que les $r_1(\overline{X}_1), \ldots, r_n(\overline{X}_n)$ sont les *sous-buts* du corps de la requête. Les jointures dans le corps de la requête sont exprimées par les multiples occurrences de la même variable. Les variables qui apparaissent dans l'entête de la requête ($\overline{X}$) sont appelées *variables distinguées* et toutes les autres variables sont des *variables existentielles* ($(\overline{X}_1 \cup \ldots \cup \overline{X}_n) - \overline{X}$). Finalement, Var(Q) désigne l'ensemble des variables de la requête Q, Subgoals(Q) désigne l'ensemble des sous-buts de Q et Q($\mathcal{D}$) désigne l'ensemble des résultats obtenus par l'exécution de Q sur une base de données $\mathcal{D}$.

Prenons par exemple la requête Q_1 de l'Exemple 4.4 et considérons que le schéma virtuel est composé des relations de l'Exemple 4.1. L'expression Datalog de Q_1 est la suivante :

Exemple 4.9 : Expression Datalog de la requête Q_1

```
Q1(idV):-
  VOYAGE(idV, prix, lieuDep, lieuArr, nbJours, typeSejour, idH, idT, idD),
  DEPART (idD, date, heure, pointRDV),
    date = '10/11/2007', lieuArr = 'Madrid', nbJours = 3.
```

L'entête de la requête Q_1 contient une seule variable distinguée `idV` qui correspond à la variable de la clause SELECT. Son corps est composé de deux sous-buts correspondant aux deux relations virtuelles sur lesquelles Q_1 est exprimée (VOYAGE et DEPART). La jointure entre ces deux sous-buts est exprimée par la répétition de la variable `idV`. Finalement, les prédicats de sélection sont ajoutés à la fin de l'expression.

Pour pouvoir illustrer le fonctionnement des algorithmes de réécriture, nous allons supposer que le schéma virtuel du système de médiation est celui de l'Exemple 4.1. Les instances de ce schéma sont calculées à partir de 7 sources de données dont les définitions sont présentées dans l'Exemple 4.10. Ces définitions seront utilisées pour réécrire la requête Q_1 de l'Exemple 4.9.

Exemple 4.10 : Exemple de définitions de sources de données

```
S1:HotelsDuMonde(idH, N, NBE, R, V) :-
    HOTEL(idH, N, NBE, R, V).
S2:TransportAerien(idT, LD, LA, D, H, M, DIR, NC):-
   TRANSPORT(idT, M, DIR, NC),
   VOYAGE(idV, P, LD, LA, NBJ, TS, idH, idT, idD),
   DEPART(idD, D, H, RDV),
        M='avion'.
S3:SNCF(idT, LD, LA, D, H, M, DIR, NC):-
   TRANSPORT(idT, M, DIR, NC),
   VOYAGE(idV, P, LD, LA, NBJ, TS, idH, idT, idD),
   DEPART(idD, D, H, RDV),
        M='train'.
S4:VoyagerPartout(idT, LD, LA, D, H, M, DIR, NC):-
   TRANSPORT(idT, M, DIR, NC),
   VOYAGE(idV, P, LD, LA, NBJ, TS, idH, idT, idD),
   DEPART(idD, D, H, RDV).
S5:PromoVacances(idV, P, LD, LA, NBJ, D, H, TS, NC, N, idT) :-
   TRANSPORT(idT, M, DIR, NC),
   VOYAGE(idV, P, LD, LA, NBJ, TS, idH, idT, idD),
   HOTEL(idH, N, NBE, R, V),
   DEPART(idD, D, H, RDV),
        LD='Paris', P<950.
S6:LyonVacances(idV, P, LD, LA, NBJ, D, H, TS, NC, N, idH, idT) :-
   TRANSPORT(idT, M, DIR, NC),
   VOYAGE(idV, P, LD, LA, NBJ, TS, idH, idT, idD),
   HOTEL(idH, N, NBE, R, V),
   DEPART(idD, D, H, RDV),
        LD='Lyon', P<950.
S7:HorairesDeparts(idD, D, H, RDV) :-
   DEPART(idD, D, H, RDV).
```

2.2.2 Algorithme des règles inversées

Le fonctionnement de l'algorithme de règles inversées est basé sur la construction d'un ensemble de règles qui montrent comment construire les tuples du schéma global à partir des tuples des vues [Qia96] [DG97]. Le principe de construction des règles inversées est le suivant :

- Pour chaque vue V_i définie par l'expression $V_i(\overline{X}) : -r_1(\overline{X}_1),...,r_n(\overline{X}_n).$, construire n règles telles que l'entête de la règle R_j, j = 1..n, est l'atome $r_j(\overline{X}_j)$ du corps de V_i et le corps de R_j est l'entête de la vue V_i ($V_i(\overline{X})$).

- remplacer toutes les variables existentielles de la définition de V_i qui se retrouvent dans les entêtes des règles inversées par des fonctions de Skolem. Utiliser la même fonction de Skolem pour toutes les occurrences de la même variable existentielle de V_i. Les fonctions de Skolem indiquent les variables dont les valeurs sont inconnues en dehors de la définition de la vue.
- L'ensemble des règles inversées est l'union des règles produites pour chaque vue.

Prenons par exemple la vue S_5 qui définit la source PROMOVACANCES. Le corps de la définition de S_5 contient quatre atomes et par conséquent, inverser S_5 revient à construire les quatre règles suivantes :

R1 : **TRANSPORT**(idT, *f1*(idV, P, LD, LA, NBJ, D, H, TS, M, N, idT), *f2*(idV, P, LD, LA, NBJ, D, H, TS, M, N, idT), NC) :-

PROMOVACANCES(idV, P, LD, LA, NBJ, D, H, TS, NC, N, idT), LD='Paris', P<950.

R2 : **VOYAGE**(idV, P, LD, LA, NBJ, TS, idT, *f3*(idV, P, LD, LA, NBJ, D, H, TS, M, N, idT), *f7*(idV, P, LD, LA, NBJ, D, H, TS, M, N, idT)) :-

PROMOVACANCES(idV, P, LD, LA, NBJ, D, H, TS, NC, N, idT), LD='Paris', P<950.

R3 : **HOTEL**(*f3*(idV, P, LD, LA, NBJ, D, H, TS, M, N, idT), N, *f4*(idV, P, LD, LA, NBJ, D, H, TS, M, N, idT), *f5*(idV, P, LD, LA, NBJ, D, H, TS, M, N, idT), *f6*(idV, P, LD, LA, NBJ, D, H, TS, M, N, idT)) :-

PROMOVACANCES(idV, P, LD, LA, NBJ, D, H, TS, NC, N, idT), LD='Paris', P<950.

R4 : **DEPART**(*f7*(idV, P, LD, LA, NBJ, D, H, TS, M, N, idT), D, H, *f8*(idV, P, LD, LA, NBJ, D, H, TS, M, N, idT) :-

PROMOVACANCES(idV, P, LD, LA, NBJ, D, H, TS, NC, N, idT), LD='Paris', P<950.

Dans la règle R2, la variable idH est une variable existentielle dans la définition de la source PROMOVACANCES. Cette variable est remplacée par la fonction de Skolem *f3*(idV, P, LD, LA, NBJ, D, H, TS, M, N, idT). La fonction de Skolem exprime le fait que pour certaines valeurs de idH, la relation VOYAGE contient un tuple de la forme (idV, P, LD, LA, NBJ, TS, idT, idH). On remarque également que la même fonction de Skolem (*f3*) est utilisée pour remplacer la même variable existentielle idH dans la règle R3. Les définitions des autres sources peuvent être inversées en suivant le même principe. Le résultat final du processus d'inversement des définitions des sources donnerait 19 règles inversées correspondant aux 19 sous-buts des requêtes de médiation. Pour simplifier l'exemple, nous nous limiterons aux 4 règles résultantes de l'inversion de la définition de la source PROMOVACANCES.

Une fois les règles inversées obtenues, la requête initiale peut être réécrite. Réécrire une requête en utilisant l'ensemble des règles inversées revient à évaluer cette requête sur les tuples produits par ces règles. Supposons par exemple que la source PROMOVACANCES possède une extension comprenant deux tuples :

{ (2, 800, 'Paris', 'Madrid', 3, '10/11/2007', 11h00, 'circuit', 3, 'El Coloso', 5),

(1, 700, 'Paris', 'Venise', 3, '11/12/2007', 21h00, 'visite', 4, 'El Coloso', 3) }

En appliquant les règles inversées produites à partir de la définition S_5 sur ces deux tuples, on obtient les tuples suivants :

Tuples produits par R4 :

{ **TRANSPORT**(5, *f1*(2, 800, ‘Paris’, ‘Madrid’, 3, ‘10/11/2007’, 11h00, ‘circuit’, 3, ‘El Coloso’, 5), *f2*(2, 800, ‘Paris’, ‘Madrid’, 3, ‘10/11/2007’, 11h00, ‘circuit’, 3, ‘El Coloso’, 5), 3) ,

TRANSPORT(3, *f1*(1, 700, ‘Paris’, ‘Venise’, 3, ‘11/12/2007’, 21h00, ‘visite’, 4, ‘El Coloso’, 3), *f2*(1, 700, ‘Paris’, ‘Venise’, 3, ‘11/12/2007’, 21h00, ‘visite’, 4, ‘El Coloso’, 3), *4*) }

Tuples produits par R2 :

{ **VOYAGE**(2, 800, ‘Paris’, ‘Madrid’, 3, ‘circuit’, 5, *f3*(2, 800, ‘Paris’, ‘Madrid’, 3, ‘10/11/2007’, 11h00, ‘circuit’, 3, ‘El Coloso’, 5, *f7*(2, 800, ‘Paris’, ‘Madrid’, 3, ‘10/11/2007’, 11h00, ‘circuit’, 3, ‘El Coloso’, 5)),

VOYAGE(1, 700, ‘Paris’, ‘Venise’, 3, ‘visite’, 3, *f3*(1, 700, ‘Paris’, ‘Venise’, 3, ‘11/12/2007’, 21h00, ‘visite’, 4, ‘El Coloso’, 3, *f7*(1, 700, ‘Paris’, ‘Venise’, 3, ‘11/12/2007’, 21h00, ‘visite’, 4, ‘El Coloso’, 3)) }

Tuples produits par R3 :

{ **HOTEL**(*f3*(2, 800, ‘Paris’, ‘Madrid’, 3, ‘10/11/2007’, 11h00, ‘circuit’, 3, ‘El Coloso’, 5), ‘El Coloso’, *f4*(2, 800, ‘Paris’, ‘Madrid’, 3, ‘10/11/2007’, 11h00, ‘circuit’, 3, ‘El Coloso’, 5), *f5*(2, 800, ‘Paris’, ‘Madrid’, 3, ‘10/11/2007’, 11h00, ‘circuit’, 3, ‘El Coloso’, 5), *f6*(2, 800, ‘Paris’, ‘Madrid’, 3, ‘10/11/2007’, 11h00, ‘circuit’, 3, ‘El Coloso’, 5),

HOTEL(*f3*(1, 700, ‘Paris’, ‘Venise’, 3, ‘11/12/2007’, 21h00, ‘visite’, 4, ‘El Coloso’, 3), ‘El Coloso’, *f4*(1, 700, ‘Paris’, ‘Venise’, 3, ‘11/12/2007’, 21h00, ‘visite’, 4, ‘El Coloso’, 3), *f5*(1, 700, ‘Paris’, ‘Venise’, 3, ‘11/12/2007’, 21h00, ‘visite’, 4, ‘El Coloso’, 3), *f6*(1, 700, ‘Paris’, ‘Venise’, 3, ‘11/12/2007’, 21h00, ‘visite’, 4, ‘El Coloso’, 3) }

Tuples produits par R4 :

{ **DEPART**(*f7*(2, 800, ‘Paris’, ‘Madrid’, 3, ‘10/11/2007’, 11h00, ‘circuit’, 3, ‘El Coloso’, 5), ‘10/11/2007’, 11h00, *f8*(2, 800, ‘Paris’, ‘Madrid’, 3, ‘10/11/2007’, 11h00, ‘circuit’, 3, ‘El Coloso’, 5)),

DEPART(*f7*(1, 700, ‘Paris’, ‘Venise’, 3, ‘11/12/2007’, 21h00, ‘visite’, 4, ‘El Coloso’, 3), ‘11/12/2007’, 21h00, *f8*(1, 700, ‘Paris’, ‘Venise’, 3, ‘11/12/2007’, 21h00, ‘visite’, 4, ‘El Coloso’, 3)) }

En exécutant la requête Q_1 de l’Exemple 4.9 sur les tuples produits par les règles R1, R2, R3 et R4 on obtient le résultat {2}.

L’avantage de l’approche de réécriture de requêtes à base de règles inversées est la construction des règles qui peut être faite en temps polynomial. Par contre le calcul des résultats de la requête peut répéter des calculs qui ont été faits pour le calcul des vues. Par exemple la définition de la vue S_5 contient la jointure entre les relations DEPART et VOYAGE et cette jointure doit être recalculée lors de l’évaluation de la requête. Un autre inconvénient de cette approche est que lors de calcul des tuples, l’algorithme va utiliser l’ensemble des règles inversées, même si certaines règles ne sont pas contributives à la récriture de la requête i.e. il n’est pas possible d’obtenir des résultats pour la requête en utilisant ces règles.

Dans notre exemple en inversant les définitions des sources, on obtient 5 règles inversées qui permettent de calculer des tuples pour la relation VOYAGE et 6 règles pour la relation DEPART ce qui donne au total 5*6 = 30 possibilités de développement de la requête Q_1 dans le pire des cas.

2.2.3 Algorithme Bucket

L’idée principale de l’algorithme Bucket est de réduire le nombre de réécritures qui doivent être étudiées en considérant chaque sous-but de la requête en isolation afin de déterminer les vues contributives pour chaque sous-but [LRO96]. L’algorithme procède en deux étapes : (i) construction d’un bucket (tas) pour chaque sous-but de la requête contenant les vues contributives pour la réécriture de ce sous-but, et (ii) construction de toutes les

requêtes conjonctives possibles en prenant une source de chaque bucket. Le résultat final est l'union de toutes les requêtes conjonctives (réécritures candidates) qui sont incluses dans la requête initiale moyennant l'ajout de prédicats de comparaison si nécessaire.

Dans sa première phase, l'algorithme Bucket essaie de trouver l'ensemble des vues qui sont contributives pour chaque sous-but de la requête. Une vue V est contributive pour un sous-but g d'une requête Q si les conditions suivantes sont vérifiées :

- V contient un sous-but g_1 tel que g peut être unifié avec g_1. L'unification est faite sur des sous-buts ayant le même nom en tenant compte de la position des variables. Elle consiste à construire les mappings entre les variables des deux sous-buts,
- Toutes les variables distinguées de Q sont mappées à des variables distinguées de V lors de l'unification de g et g_1,
- Les prédicats de comparaisons de V et de Q ne sont pas conflictuels. Plus de détails sur la compatibilité des prédicats de la requête et des vues peuvent être trouvés dans la section suivante qui décrit l'algorithme MiniCon.

Dans la phase de construction des buckets, si un sous-but g de la requête Q s'unifie avec plusieurs sous-buts d'une vue V, alors le bucket correspondant à g contient plusieurs occurrences de V.

Prenons par exemple la requête Q_1 de l'Exemple 4.9 et les définitions des sources de l'Exemple 4.10. La requête Q_1 contient deux sous-buts et par conséquent lors de sa première phase, l'algorithme va créer les deux buckets correspondants. Le premier sous-but de Q_1 (VOYAGE(idV, prix, lieuDep, lieuArr, nbJours, typeSejour, idH, idT, idD)) peut être unifié avec les sous-buts du même nom des sources TRANSPORTAERIEN, SNCF, VOYAGERPARTOUT, PROMOVACANCES et LYONVACANCES. Le résultat de cette unification est un ensemble de mappings entre les variables de la requête et celles des sources. Par exemple les mappings de l'unification des sous-buts nommés VOYAGE dans Q_1 et dans PROMOVACANCES sont :
{ idV→idV, prix→P, lieuDep→LD, lieuArr→LA, nbJours→NBJ, typeSejour→TS, idH→idH, idT→idT, idD→idD }

En construisant les mappings entre les variables des sous-buts de la requêtes et ceux des définitions des sources, l'algorithme va détecter que la variable distinguée idV de la requête est mappée à une variable existentielle dans les 3 premières sources. Par conséquent, TRANSPORTAERIEN, SNCF et VOYAGERPARTOUT ne sont pas ajoutées au bucket du premier sous-but de Q_1. Les deux sources restantes (PROMOVACANCES et LYONVACANCES) vérifient les conditions de construction des tas et sont ajoutées au bucket de VOYAGE. Lors de l'unification du second sous-but de la requête Q_1 (DEPART), l'algorithme détecte que les sources TRANSPORTAERIEN, SNCF, VOYAGERPARTOUT, PROMOVACANCES, LYONVACANCES et HORAIRESDEPART contiennent toutes un sous-but pouvant être unifié avec celui de Q_1. Ces sources composent le bucket de ce sous-but. Le résultat final de la construction des buckets est présenté dans le Tableau 4.1. Notons que les noms des variables de l'entête de chaque source qui apparaissent dans les mappings sont remplacés par les noms des variables de la requête (par exemple P est remplacée par *prix* dans la source *PromoVacances*).

VOYAGE(idV, prix, lieuDep, lieuArr, nbJours, typeSejour, idH, idT)	DEPART(idD, date, heure, pointRDV)
PROMOVACANCES(idV, prix, lieuDep, lieuArr, nbJours, D, H, typeSejour, NC, N, idT) **LYONVACANCES**(idV, prix, lieuDep, lieuArr, nbJours, D, H, typeSejour, NC, N, idH, idT)	**TRANSPORTAERIEN**(idT, LD, LA, date, heure, M, DIR, NC), **SNCF**(idT, LD, LA, date, heure, M, DIR, NC), **VOYAGERPARTOUT**(idT, LD, LA, date, heure, M, DIR, NC), **PROMOVACANCES**(idV, P, LD, LA, NBJ, date, heure, TS, NC, N, idT) **LYONVACANCES**(idV, P, LD, LA, NBJ, date, heure, TS, NC, N, idH, idT) **HORAIRESDEPARTS**(idD, date, heure, pointRDV)

Tableau 4.1 : Buckets des sous-buts de la requête Q_1

La deuxième étape de l'algorithme consiste à vérifier toutes les combinaisons de sources possibles obtenues à partir du produit cartésien des sources des deux buckets.

Dans notre exemple il y a 2*6=12 combinaisons à tester :

(1) Q'_1(idV) :- **PROMOVACANCES**(idV, prix, lieuDep, lieuArr, nbJours, D, H, typeSejour, NC, N, idT), **TRANSPORTAERIEN**(idT, LD, LA, date, heure, M, DIR, NC).

(2) Q'_1(idV) :- **PROMOVACANCES**(idV, prix, lieuDep, lieuArr, nbJours, D, H, typeSejour, NC, N, idT), **SNCF**(idT, LD, LA, date, heure, M, DIR, NC).

(3) Q'_1(idV) :- **PROMOVACANCES**(idV, prix, lieuDep, lieuArr, nbJours, D, H, typeSejour, NC, N, idT), **VOYAGERPARTOUT**(idT, LD, LA, date, heure, M, DIR, NC).

(4) Q'_1(idV) :- **PROMOVACANCES**(idV, prix, lieuDep, lieuArr, nbJours, D, H, typeSejour, NC, N, idT), **PROMOVACANCES**(idV, P, LD, LA, NBJ, date, heure, TS, NC, N, idT).

(5) Q'_1(idV) :- **PROMOVACANCES**(idV, prix, lieuDep, lieuArr, nbJours, D, H, typeSejour, NC, N, idT), **LYONVACANCES**(idV, P, LD, LA, NBJ, date, heure, TS, NC, N, idH, idT).

(6) Q'_1(idV) :- **PROMOVACANCES**(idV, prix, lieuDep, lieuArr, nbJours, D, H, typeSejour, NC, N, idT), **HORAIRESDEPARTS**(idD, date, heure, pointRDV).

(7) Q'_1(idV) :- **LYONVACANCES**(idV, prix, lieuDep, lieuArr, nbJours, D, H, typeSejour, NC, N, idH, idT), **TRANSPORTAERIEN**(idT, LD, LA, date, heure, M, DIR, NC).

(8) Q'_1(idV) :- **LYONVACANCES**(idV, prix, lieuDep, lieuArr, nbJours, D, H, typeSejour, NC, N, idH, idT), **SNCF**(idT, LD, LA, date, heure, M, DIR, NC).

(9) Q'_1(idV) :- **LYONVACANCES**(idV, prix, lieuDep, lieuArr, nbJours, D, H, typeSejour, NC, N, idH, idT), **VOYAGERPARTOUT**(idT, LD, LA, date, heure, M, DIR, NC).

(10) Q'_1(idV) :- **LYONVACANCES**(idV, prix, lieuDep, lieuArr, nbJours, D, H, typeSejour, NC, N, idH, idT), **PROMOVACANCES**(idV, P, LD, LA, NBJ, date, heure, TS, NC, N, idT).

(11) Q'_1(idV) :- **LYONVACANCES**(idV, prix, lieuDep, lieuArr, nbJours, D, H, typeSejour, NC, N, idH, idT), **LYONVACANCES**(idV, P, LD, LA, NBJ, date, heure, TS, NC, N, idH, idT).

(12) Q'_1(idV) :- **LYONVACANCES**(idV, prix, lieuDep, lieuArr, nbJours, D, H, typeSejour, NC, N, idH, idT), **HORAIRESDEPARTS**(idD, date, heure, pointRDV).

Les multiples occurrences d'une source dans la même expression sont supprimées en faisant l'union des noms des variables qui apparaissent dans les mappings. Par exemple l'expression (4) est équivalente à :

Q'_1(idV) **:- PROMOVACANCES**(idV, prix, lieuDep, lieuArr, nbJours, date, heure, typeSejour, NC, N, idT).

En prenant en compte la jointure entre les deux sous-buts de la requête Q_1 sur l'attribut idD, on s'aperçoit que dans les sources TRANSPORTAERIEN, SNCF et VOYAGERPARTOUT cet attribut est mappé à une variable existentielle. Étant donné que les définitions de ces sources ne vérifient pas cette jointure et qu'il est impossible de la faire avec une autre source (idD est une variable existentielle), toutes les solutions potentielles contenant une de ces sources ne peuvent pas former une réécriture. Ceci élimine les candidats (1), (2), (3), (7), (8) et (9). En regardant les sources PROMOVACANCES et LYONVACANCES, on peut voir que la variable de jointure (idD) de Q_1 est également mappée à une variable existentielle. La différence dans ce cas est que les définitions de ces sources incluent la jointure. Autrement dit, il faut utiliser la même source pour couvrir les deux sous-buts de la requête afin que la jointure soit satisfaite ce qui exclue les solutions (5), (6), (10) et (12). Les solutions candidates qui restent (4) et (11), forment des réécritures candidates parce qu'elles vérifient la jointure et couvrent tous les sous-buts de la requête. Le résultat final de la réécriture de Q_1 est l'union des candidates (4) et (11) auxquelles sont ajoutés les prédicats de la requête (Exemple 4.11).

Exemple 4.11 : Résultat de la réécriture de la requête initiale Q_1

Q'_1(idV) :-

PROMOVACANCES(idV, prix, lieuDep, lieuArr, nbJours, date, heure, typeSejour, NC, N, idT), date = '10/11/2007', lieuArr = 'Madrid', nbJours = 3

∪

LYONVACANCES(idV, prix, lieuDep, lieuArr, nbJours, date, heure, typeSejour, NC, N, idT, idH), date = '10/11/2007', lieuArr = 'Madrid', nbJours = 3.

Pour éviter de tester toutes les combinaisons de sources obtenues par le produit cartésien, certaines approches proposent de considérer les combinaisons de sources comme des itemsets fréquents et de calculer leur bordure positive et négative [JPR+05].

L'inconvénient majeur de l'algorithme Bucket est son incapacité de détecter le fait qu'il doit utiliser la même vue pour couvrir plusieurs sous-buts de la requête. Cet algorithme est également incapable de trouver les vues qui ne peuvent pas être contributives (comme par exemple la source TRANSPORTAERIEN) avant d'avoir testé toutes les solutions possibles. La section suivante présente l'algorithme MiniCon qui présente des améliorations qui permettent de gommer les inconvénients de l'algorithme Bucket.

2.2.4 Algorithme MiniCon

Comme l'algorithme Bucket, l'algorithme MiniCon [HP01] fonctionne en deux étapes qui sont la recherche de vues contributives pour la réécriture de la requête et la combinaison de ces vues afin d'obtenir les réécritures de la requête. Dans la première phase, il cherche des correspondances entre les sous-buts de la requête et ceux des vues. À la différence de l'algorithme Bucket qui construit des tas indépendants pour chaque sous-but, lorsque l'algorithme MiniCon détecte une correspondance entre un sous-but *g* de la requête Q et un sous-but g_1 d'une vue V, il examine les variables de jointure de la requête afin de déterminer l'ensemble minimal de sous-buts de V qui doivent être unifiés avec des sous-buts de Q étant donné que *g* sera unifié avec g_1. Autrement dit, l'algorithme vérifie que toutes les jointures de Q sont soit vérifiées par la définition de la vue, soit exprimées sur des variables distinguées de la vue afin d'être satisfaites ultérieurement. Pour chaque vue, l'algorithme note les sous-buts de la requête qui peuvent être couverts par cette vue. L'ensemble des informations décrivant les mappings entre une vue V et une requête Q sont représentées par un quadruplet $(h, V(\overline{Y}), \varphi, G)$ appelé MiniCon Descriptor (MCD) où :

- *h* est un homomorphisme défini sur les variables de l'entête de V. Elle exprime le fait que parfois il peut être nécessaire d'unifier des variables de l'entête de la vue.
- $V(\overline{Y})$ est le résultat de l'application de *h* sur l'entête de V.
- φ est le mapping (unification) partiel des variables de Vars(Q) vers Vars(V).
- *G* est l'ensemble de sous-buts de Q qui sont couverts par la vue V.

En créant les MCDs pour la requête Q_1 de l'Exemple 4.9 et les définitions des source de l'Exemple 4.10, l'algorithme MiniCon va détecter l'impossibilité des sources TRANSPORTAERIEN, SNCF et VOYAGERPARTOUT de satisfaire la jointure de Q_1. Aucun MCD ne sera créé pour ces sources. L'algorithme va également s'apercevoir que les sources PROMOVACANCES et LYONVACANCES doivent être utilisées pour couvrir les deux sous-buts de la requête. Finalement, la source HORAIRESDEPARTS peut couvrir le second sous-but de Q_1. Les MCDs générés dans ce cas sont présentés dans le Tableau 4.2. Étant donné qu'aucune variable de l'entête des sources ne doit être unifiée, *h* est la fonction d'identité. Elle n'est pas présente dans le tableau afin de ne pas le surcharger.

$V(\overline{Y})$	φ	G
PROMOVACANCES(idV, P, LD, LA, NBJ, D, H, TS, NC, N, idT)	idV→idV, prix→P, lieuDep→LD, lieuArr→LA, nbJours→NBJ, typeSejour→TS, idH→idH, idT→idT, idD→idD, date→D, heure→H	1, 2
LYONVACANCES(idV, P, LD, LA, NBJ, D, H, TS, NC, N, idT, idH)	idV→idV, prix→P, lieuDep→LD, lieuArr→LA, nbJours→NBJ, typeSejour→TS, idH→idH, idT→idT, idD→idD, date→D, heure→H	1, 2
HORAIRESDEPARTS(idD, D, H, RDV)	idD→idD, date→D, heure→H, pointRDV→RDV	2

Tableau 4.2 : MCDs générés pour Q_1

Une fois les MCDs construits, l'algorithme les combine afin d'obtenir des réécritures de la requête. La combinaison de MCDs est faite en suivant le principe que l'ensemble des

MCDs formant une récriture candidate de la requête doivent couvrir l'ensemble des sous-buts de la requête et que chaque paire de MCDs doivent courir des sous-buts disjoints i.e. :

Soient C_1, C_2, …, C_k un ensemble de MCDs formant une réécriture candidate de Q. Alors les deux conditions suivantes doivent être vérifiées :

- $G_1 \cup \ldots \cup G_k = \text{Subgoals}(Q)$ et
- $\forall i, j, i \neq j, G_i \cap G_j = \varnothing$

Pour l'exemple du Tableau 4.2, les deux réécriture candidates de Q_1 qui vérifient ces conditions sont faites avec les MCDs des sources PROMOVACANCES et LYONVACANCES. Le résultat de la réécriture est le même que celui obtenu par l'algorithme Bucket (Exemple 4.11) :

Q'_1(idV) :-

PROMOVACANCES(idV, prix, lieuDep, lieuArr, nbJours, date, heure, typeSejour, NC, N, idT), date = '10/11/2007', lieuArr = 'Madrid', nbJours = 3

∪

LYONVACANCES(idV, prix, lieuDep, lieuArr, nbJours, date, heure, typeSejour, NC, N, idT, idH), date = '10/11/2007', lieuArr = 'Madrid', nbJours = 3.

Une propriété très intéressante de MiniCon est le fait qu'il vérifie la satisfaisabilité des prédicats de la requête ainsi que la compatibilité des MCDs. Lorsque les vues et la requête contiennent des prédicats de sélection, l'algorithme garantit que les réécritures produites sont correctes et lorsque les prédicats sont sous la forme x θ c où *x* est une variable, *c* est une constante et $\theta \in \{<, >, \leq, \geq\}$, MiniCon produit des réécritures maximalement-incluses dans la requête.

Le principe de vérification de la satisfaisabilité des prédicats de sélection de la requête est proche de celui de la vérification des jointures. Lorsqu'un prédicat de sélection $x\ \theta\ c$ d'une requête Q est exprimé sur un attribut *x* qui apparaît dans un sous-but couvert par une vue V, la vue peut être contributive si une des conditions suivantes est satisfaite :

- la définition de V, satisfait le prédicat ou
- x est mappé à une variable distinguée dans V et de ce fait le prédicat peut être exprimé sur V.

De la même manière, pendant la phase de combinaison des MCDs, toute solution potentielle contenant des MCDs dont les définitions sont conflictuelles, sont éliminées. Les conflits entre les définitions de deux MCDs sont détectés à partir des prédicats exprimés sur des variables des MCDs mappées à la même variable de la requête.

Le principal avantage de l'algorithme MiniCon est sa capacité de filtrer les vues non contributives pendant la phase de construction des MCDs. Les MCDs construits peuvent être combinés sans vérifier la satisfaisabilité des prédicats de jointure. Les seules vérifications qui restent à faire visent à détecter des redondances et s'assurer que tous les sous-buts de la requête sont couverts.

3. Approches de reformulation de requêtes

Cette section décrit et analyse deux approches de reformulation de requêtes qui résultent de la composition des techniques de réécriture et d'enrichissement.

Soit $\mathcal{R}(Q / S_m)$ l'algorithme de réécriture $\mathcal{R}$ d'une requête Q sur les définitions d'un ensemble de sources $S_m = \{S_1, \ldots, S_p\}$ et soit $\mathcal{E}(Q_i / [P_u, S_v])$ l'algorithme d'enrichissement $\mathcal{E}$

d'une requête Q, qui est exprimée sur un schéma S_v, avec le profil utilisateur P_u. Selon l'ordre dans lequel sont appliqués ces algorithmes, on obtient deux approches de reformulation personnalisée de requêtes : une approche réécriture-enreichissement $\mathcal{E}(\mathcal{R})$ et une approche enrichissement-réécriture $\mathcal{R}(\mathcal{E})$.

3.1 Approche enrichissement – réécriture

L'approche enrichissement-réécriture, notée $\mathcal{R}(\mathcal{E})$, consiste à enrichir d'abord la requête utilisateur à l'aide de son profil, sans tenir compte des sources de données et à rechercher ensuite les réécritures possibles de la requête enrichie (Figure 4.6). On peut formaliser cette approche comme suit :

$\mathcal{R}(\mathcal{E}(Q_u / [P_u,S_v]) / S_m)$

$\mathcal{E}(Q_u / [P_u,S_v])$ délivre l'enrichissement Q_{u+} de la requête Q_u par rapport au profil utilisateur P_u et au schéma virtuel S_v. $\mathcal{R}(Q_{u+} / S_m)$ est la réécriture de la requête enrichie Q_{u+} par rapport à l'ensemble S_m des définitions LAV des sources de données.

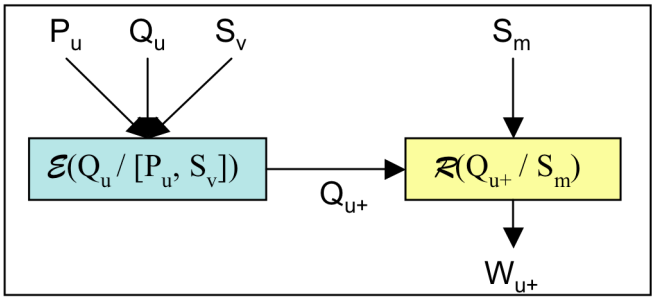


Figure 4.6 : Approche Enrichissement-Rééecriture

Avant de discuter les avantages et les inconvénients de l'approche $\mathcal{R}(\mathcal{E})$, il faut montrer que les requêtes produites par l'algorithme d'enrichissement peuvent être utilisées par l'algorithme de réécriture. Comme le montre l'Exemple 4.7, la requête enrichie peut contenir une disjonction de prédicats qui ne peut pas être traitée par les algorithmes de réécriture. Afin d'obtenir des requêtes conjonctives, la requête enrichie doit être décomposée en plusieurs requêtes conjonctives, chacune contenant un terme de la disjonction. Chaque requête conjonctive ainsi obtenue peut être réécrite séparément.

Prenons par exemple le profil P_1 de l'Exemple 4.2, la requête utilisateur Q_1 de l'Exemple 4.4 et les valeurs des paramètres de l'algorithme d'enrichissement de la section 2.1 (K=5, M=2, L=2). La requête enrichie qu'on obtient est la requête Q_{1+} de l'Exemple 4.7 qui peut être décomposée en 3 requêtes conjonctives dont les expressions Datalog sont présentées dans l'Exemple 4.12.

Exemple 4.12 : Décomposition de la requête enrichies Q_{1+}

```
Q¹1+(idV):-
   VOYAGE(idV, prix, lieuDep, lieuArr, nbJours, typeSejour, idH, idT, idD),
   DEPART (idD, date, heure, pointRDV),
   TRANSPORT(idT, moyen, direct, niveauConfort),
      date = '10/11/2007', lieuArr = 'Madrid', nbJours = 3,
      lieuDep = 'Toulouse', moyen='avion', heure > 22hOO, direct=VRAI.
Q²1+(idV):-
   VOYAGE(idV, prix, lieuDep, lieuArr, nbJours, typeSejour, idH, idT, idD),
   DEPART (idD, date, heure, pointRDV),
   TRANSPORT(idT, moyen, direct, niveauConfort),
      date = '10/11/2007', lieuArr = 'Madrid', nbJours = 3,
      lieuDep = 'Toulouse', moyen='avion', heure > 22hOO, prix<1000.
```

```
Q³1+(idV):-
   VOYAGE(idV, prix, lieuDep, lieuArr, nbJours, typeSejour, idH, idT, idD),
   DEPART (idD, date, heure, pointRDV),
   TRANSPORT(idT, moyen, direct, niveauConfort),
      date = '10/11/2007', lieuArr = 'Madrid', nbJours = 3,
      lieuDep = 'Toulouse', moyen = 'avion', direct=VRAI, prix < 1000.
```

L'avantage de l'approche $\mathcal{R}(\mathcal{E})$ est qu'elle permet d'enrichir la requête initiale avec tous les prédicats du profil utilisateur qui ne sont pas conflictuels avec celle-ci. Ceci permet d'exploiter au maximum le profil utilisateur. Par contre, étant donné que le processus d'enrichissement ajoute des prédicats à la requête sans tenir compte des définitions des sources de données, elle comporte un certain nombre de risques.

Le premier risque est que la requête enrichie peut être trop complexe à réécrire. Il vient du fait que pendant la phase d'enrichissement de nouvelles relations virtuelles sont ajoutées à la requête. Dans ce cas, il y a un risque que la requête enrichie contienne trop de relations ce qui peut engendrer une réécriture trop coûteuse ou encore que le nombre de réécritures candidates soit trop important.

Le second risque est que certains des prédicats ajoutés pendant la phase d'enrichissement soient :

- contradictoires avec les définitions des sources et éliminer des sources pertinentes ou dans le cas extrême éliminer toutes les réécritures candidates,
- satisfaits par les définitions des sources de toutes les réécritures candidates. Dans ce cas l'ajout de ces prédicats à la requête ne modifie pas les résultats des reformulations produites par l'approche.

Prenons la requête Q^2_{1+} de l'Exemple 4.12 (les requêtes Q^2_{1+} et Q^3_{1+} se réécrivent de la même manière) et les définitions des sources de données de l'Exemple 4.10. Comparée à la requête initiale Q_1, elle contient un nouveau sous-but TRANSPORT et 4 nouveaux prédicats de sélection : « lieuDep = 'Toulouse' », « moyen='avion' », « heure > 22hOO » et « prix<1000 ». Les MCDs pouvant être générés pour la réécriture de cette requête sont présentés dans le Tableau 4.3. Le premier constat est la disparition des deux MCDs venant des sources PROMOVACANCES et LYONVACANCES. Ceci est dû au prédicat « lieuDep = 'Toulouse' » qui est conflictuel avec la définition de ces deux sources. Le second constat est l'apparition de deux nouveaux MCDs qui couvrent le sous-but TRANSPORT. Le sous-but TRANSPORT peut être unifié à un des sous-but dans les sources TRANSPORTAERIEN, SNCF, VOYAGERPARTOUT, PROMOVACANCES et LYONVACANCES, mais seules les 3 premières sources permettent d'exprimer le prédicat moyen='avion' sur leur schéma et la définition de la source SNCF est conflictuelle avec lui. Cette dernière source ne peut donc pas être utilisée pour couvrir le sous-but TRANSPORT. En utilisant ces MCDs, il n'est pas possible de réécrire la requête Q^2_{1+}, parce qu'il est impossible de couvrir le sous-but VOYAGE. En effet, lors du processus d'enrichissement de la requête Q_1, le prédicat « lieuDep = 'Toulouse' » a été ajouté à celle-ci et ce prédicat élimine les sources PROMOVACANCES et LYONVACANCES qui sont les seules à pouvoir couvrir le sous-but VOYAGE. Dans ce cas le prédicat « lieuDep = 'Toulouse' » ne peut pas être réécrit.

Définition 4.5 : Un prédicat p d'un profil utilisateur P ne peut pas être réécrit dans une requête Q si p est conflictuel avec la définition de toutes les sources pouvant couvrir le sous-but contenant l'attribut sur lequel p est exprimé.

$V(\overline{Y})$	φ	G
HORAIRESDEPARTS(idD, D, H, RDV)	idD→idD, date→D, heure→H, pointRDV→RDV	2
TRANSPORTAERIEN(idT, LD, LA, D, H, M, DIR, NC)	idT→idT, moyen→M, direct→DIR, niveauConfort→NC	3
VOYAGERPARTOUT(idT, LD, LA, D, H, M, DIR, NC)	idT→idT, moyen→M, direct→DIR, niveauConfort→NC	3

Tableau 4.3 : MCDs générés pour la requête Q^2_{1+}

Pour pouvoir obtenir des réécritures candidates à l'enrichissement de la requête Q_1 il ne faut pas utiliser le prédicat lieuDep = 'Toulouse' dans la phase d'enrichissement. Prenons la requête Q^2_{1+} sans ce prédicat Exemple 4.13.

Exemple 4.13 : Requête enrichie Q^2_{1+} sans le prédicat « lieuDep = 'Toulouse' »

Q^2_{1+}(idV):-
VOYAGE(idV, prix, lieuDep, lieuArr, nbJours, typeSejour, idH, idT, idD),
DEPART (idD, date, heure, pointRDV),
TRANSPORT(idT, moyen, direct, niveauConfort),
date = '10/11/2007', lieuArr = 'Madrid', nbJours = 3,
moyen='avion', heure > 22hOO, prix<1000.

$V(\overline{Y})$	φ	G
HORAIRESDEPARTS(idD, D, H, RDV)	idD→idD, date→D, heure→H, pointRDV→RDV	2
TRANSPORTAERIEN(idT, LD, LA, D, H, M, DIR, NC)	idT→idT, moyen→M, direct→DIR, niveauConfort→NC	3
VOYAGERPARTOUT(idT, LD, LA, D, H, M, DIR, NC)	idT→idT, moyen→M, direct→DIR, niveauConfort→NC	3
PROMOVACANCES(idV, P, LD, LA, NBJ, D, H, TS, NC, N, idT)	idV→idV, prix→P, lieuDep→LD, lieuArr→LA, nbJours→NBJ, typeSejour→TS, idH→idH, idT→idT, idD→idD, date→D, heure→H	1, 2
LYONVACANCES(idV, P, LD, LA, NBJ, D, H, TS, NC, N, idT, idH)	idV→idV, prix→P, lieuDep→LD, lieuArr→LA, nbJours→NBJ, typeSejour→TS, idH→idH, idT→idT, idD→idD, date→D, heure→H	1, 2

Tableau 4.4 : MCDs générés pour la requête Q^2_{1+} sans le prédicat « lieuDep = 'Toulouse' »

Pour cette requête l'algorithme MiniCon peut générer les MCDs venant des sources PROMOVACANCES et LYONVACANCES Tableau 4.4. Le résultat de la réécriture de Q^2_{1+} est l'union de quatre réécritures candidates:

Q'^2_{1+}(idV) :-

PROMOVACANCES(idV, prix, lieuDep, lieuArr, nbJours, date, heure, typeSejour, NC, N, idT), **TRANSPORTAERIEN**(idT, LD, LA, D, H, moyen, direct, niveauConfort), date = '10/11/2007', lieuArr = 'Madrid', nbJours = 3, moyen='avion', heure>22hOO, prix<1000.

∪

LYONVACANCES(idV, prix, lieuDep, lieuArr, nbJours, date, heure, typeSejour, NC, N, idT, idH), **TRANSPORTAERIEN**(idT, LD, LA, D, H, moyen, direct,

niveauConfort), date = '10/11/2007', lieuArr = 'Madrid', nbJours = 3, moyen='avion', heure>22h00, prix<1000.

∪

PROMOVACANCES(idV, prix, lieuDep, lieuArr, nbJours, date, heure, typeSejour, NC, N, idT), **VOYAGERPARTOUT**(idT, LD, LA, D, H, moyen, direct, niveauConfort), date = '10/11/2007', lieuArr = 'Madrid', nbJours = 3, moyen='avion', heure>22h00, prix<1000.

∪

LYONVACANCES(idV, prix, lieuDep, lieuArr, nbJours, date, heure, typeSejour, NC, N, idT, idH), **VOYAGERPARTOUT**(idT, LD, LA, D, H, moyen, direct, niveauConfort), date = '10/11/2007', lieuArr = 'Madrid', nbJours = 3, moyen='avion', heure>22h00, prix<1000.

En analysant ce résultat, on peut s'apercevoir que le nombre de réécritures candidates a doublé par rapport à la requête initiale. Ceci est la conséquence de l'ajout du nouveau sous-but TRANSPORT à la requête initiale pendant la phase d'enrichissement. Un autre constat intéressant est que les définitions des sources utilisées pour l'ensemble des réécritures candidates satisfont le prédicat (j) (« prix<1000 ») du profil utilisateur P_1 parce que PROMOVACANCES et LYONVACANCES contiennent toutes les deux le prédicat « P<950 » qui est plus restrictif que (j) et il y a un mapping entre les attributs *P* et *prix*. Autrement dit, le prédicat « prix<1000 » est *inutile* pour l'enrichissement de la requête Q_1.

Définition 4.6 : Un prédicat p d'un profil utilisateur P est inutile à l'enrichissement d'une requête Q si p est inclus dans la définition des sources de toutes les réécritures candidates de Q.

Comme décrit dans la section 2.1 , l'algorithme d'enrichissement est combinatoire en fonction des prédicats du profil utilisateur et il est souvent nécessaire de borner leur nombre grâce au paramètre Top K. Éviter de choisir des prédicats inutiles parmi les Top K permet de prendre en compte davantage de prédicats du profil utilisateur sans en choisir plus (la pertinence des résultats augmente pour la même complexité de calcul de l'algorithme d'enrichissement).

Les inconvénients de l'approche enrichissement-réécriture sont dus principalement au fait que l'enrichissement de la requête initiale ne tient pas compte des définitions des sources de données. Une solution possible à ce problème est de prendre en compte les contraintes sur le contenu des sources avant de choisir les prédicats du profil utilisateur qui vont enrichir sa requête. La section suivante décrit une telle approche.

3.2 Approche réécriture – enrichissement

L'idée principale de l'approche réécriture-enrichissement ($\mathcal{E}(\mathcal{R})$) est d'effectuer la réécriture de la requête en premier, ce qui permet d'obtenir plusieurs réécritures candidates qui contiennent les prédicats de sélection des sources qu'elles interrogent en plus des prédicats de la requête utilisateur. Ensuite, chacune des réécritures est enrichie à l'aide du profil utilisateur (Figure 4.7). Cette approche peut être formalisée par l'expression suivante :

$$\mathcal{E}(\mathcal{R}(Q_u / S_m) / [P_u, S_m])$$

$\mathcal{R}(Q_u / S_m)$ délivre l'ensemble des réécritures candidates W_u de la requête Q_u par rapport à l'ensemble S_m des définitions LAV des sources de données. $\mathcal{E}(W_u / [P_u, S_m])$ délivre les enrichissements de l'ensemble W_u de réécritures de Q_u avec le profil P_u.

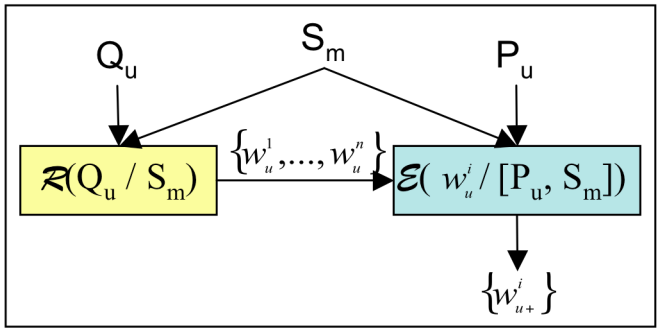


Figure 4.7 : Approche Réécriture-Enrichissement

Pour pouvoir enrichir les réécritures candidates avec les prédicats du profil utilisateur, il faut régler le problème suivant : les prédicats du profil utilisateur sont exprimés sur les attributs des relations du schéma virtuel alors que l'algorithme d'enrichissement est appliqué sur les réécritures candidates de la requête initiale qui sont exprimées sur les schéma des sources de données. Pour pouvoir enrichir les réécritures candidates d'une requête, nous utilisons les mappings entre les attributs du schéma virtuel et les attributs des sources construits par l'algorithme de réécriture. Pour chaque prédicat du profil utilisateur sous la forme « $R.x\ \theta\ c$ » où x est l'attribut sur lequel le prédicat est exprimé, R est la relation virtuelle à laquelle appartient x, θ est un opérateur et c une constante, il faut :

- identifier les MCDs qui couvrent le sous-but R de la requête,
- trouver les attributs y_i de ces MCDs pour lesquels il existe un mapping $x \rightarrow y_i$ dans φ,
- remplacer R.x avec y_i dans l'expression du prédicat $R.x\ \theta\ c$ qui devient $y_i\ \theta\ c$,
- vérifier que $y_i\ \theta\ c$ n'est pas conflictuel avec la définition de la source pour laquelle le MCD a été généré et que $y_i\ \theta\ c$ peut être exprimé sur la source (soit y_i est une variable distinguée, soit $y_i\ \theta\ c$ est inclut dans la définition de la source).

Si l'attribut x est mappé à plusieurs attributs y_i, il faut que tous les prédicats transformés $y_i\ \theta\ c$ soient non conflictuels avec les définitions des sources qui contiennent les attributs y_i et il suffit qu'un seul de ces prédicats puisse être exprimé sur les sources d'une même réécriture candidate. Ceci est la conséquence du fait que nous traitons des requêtes conjonctives.

Prenons par exemple le prédicat « DEPART.heure > 22hOO » du profil utilisateur P_1 (Exemple 4.2) et la requête initiale Q_1 (Exemple 4.4). Pour cette requête l'algorithme de réécriture peut générer 3 MCDs (Tableau 4.2), 2 desquels couvrent le sous but DEPART. Ces 2 MCDs sont générés pour les sources PROMOVACANCES et LYONVACANCES et il y a un mapping entre l'attribut heure de Q_1 et l'attribut H des deux sources (heure→H). Le prédicat « H > 22h00 » n'est pas conflictuel avec la définition de PROMOVACANCES et LYONVACANCES et l'attribut H est un attribut distingué dans les 2 sources ce qui fait que ce prédicat peut être exprimé sur toute réécriture candidate qui est faite avec une des deux sources. Remarquons que nous avons pris les noms des attributs du schéma virtuel comme noms des attributs de la requête initiale. Ceci permet d'obtenir des réécritures candidates dans lesquelles il est facile de distinguer les attributs des sources mappés à des attributs du schéma virtuel. Prenons par exemple la réécriture candidate de Q_1 qui est faite sur la source PromoVacances de l'Exemple 4.11 :

PROMOVACANCES(idV, prix, lieuDep, lieuArr, nbJours, date, heure, typeSejour, NC, N, idT), date = '10/11/2007', lieuArr = 'Madrid', nbJours = 3.

Dans cette expression les seuls attributs de PROMOVACANCES pour lesquels il n'y a pas de mappings avec un attribut de la requête sont NC et N. Tous les autres attributs apparaissent avec leurs noms dans la requête qui sont les mêmes que ceux du schéma virtuel.

Cette dernière remarque permet d'utiliser les prédicats du profil utilisateur pour enrichir les réécritures candidates d'une requête sans avoir à transformer ces prédicats.

Le principal avantage de l'approche réécriture-enrichissement est qu'elle tient compte des définitions des sources dans le choix des prédicats du profil utilisateur qui sont utilisés dans la phase d'enrichissement. Autrement dit, elle est capable de détecter les prédicats du profil utilisateur qui ne peuvent pas être réécrits ou qui sont inutiles. Les inconvénients de l'approche $\mathcal{E}(\mathcal{R})$ sont :

- elle ne tient pas compte des prédicats conflictuels avec les sources. Le choix des prédicats du profil qui peuvent enrichir une réécriture candidate est fait selon les prédicats de cette réécriture ce qui inclut ceux des définitions des sources,
- la phase de réécriture de la requête fixe le schéma. Par conséquent, l'approche $\mathcal{E}(\mathcal{R})$ ne peut utiliser que les prédicats du profil utilisateur qui sont exprimables sur les schémas des sources des réécritures candidates.

Prenons par exemple les réécritures candidates de la requête initiale Q_1 (Exemple 4.11), le profil utilisateur P_1 (Exemple 4.2) et les valeurs K=5, M=2 et L=2 pour l'algorithme d'enrichissement. Le prédicat (e) est conflictuel avec les 2 réécritures candidates car il est conflictuel avec la requête initiale. En analysant les prédicats dans les réécritures candidates, il est possible de détecter que le prédicat « VOYAGE.lieuDep = 'Toulouse' », qui ne peut pas être réécrit, est conflictuel avec les définitions des sources des deux réécritures candidates et que le prédicat « VOYAGE.prix < 1000 » est inclus dans leurs définitions. Par conséquent ces deux prédicats ne seront pas utilisés pendant la phase d'enrichissement. Les prédicats (g), (i), (k), (m), (o) et (p) sont exprimés sur des attributs des relations virtuelles TRANSPORT et HOTEL et l'algorithme de réécriture n'a pas construit de mappings pour les attributs de ces relations. Par conséquent, ces prédicats ne peuvent pas être utilisés pour enrichir les réécritures candidates de la requête Q_1.

Nous avons vu que l'algorithme d'enrichissement peut ajouter de nouvelles relations virtuelles à la requête initiale s'il y a des prédicats qui sont exprimés sur leurs attributs. Lorsque la réécriture est faite en premier, ceci n'est plus possible pace que les réécritures candidates sont exprimées sur les schémas des sources de données et non sur le schéma virtuel.

Un cas très intéressant représente le prédicat « VOYAGE.lieuDep = 'Paris' » du profil utilisateur P_1. Il est conflictuel avec la définition de la source LYONVACANCES qui est utilisée pour la seconde réécriture candidate de Q_1 et est inclus dans la définition de la source PROMOVACANCES qui est utilisée pour la première réécriture candidate. Ce prédicat ne sera utilisé pour enrichir aucune des deux réécritures candidates. Ce type de prédicats peut jouer le rôle de filtres car ils ne peuvent pas servir à enrichir les réécritures d'une requête, mais peuvent être utilisés pour éliminer celles qui ne les satisfont pas (par exemple éliminer la réécriture candidate de Q_1 qui est faite avec la source LYONVACANCES).

Définition 4.7 : Un prédicat *filtre* est un prédicat du profil utilisateur qui est soit conflictuel, soit inclus dans la définition des sources de toutes les réécritures candidates d'une requête.

Les prédicats restants du profil utilisateur ((h) et (l)) sont utilisés pour enrichir les deux réécritures candidates, mais la première est plus pertinente que la seconde car sa définition inclut en plus le prédicat (n) du profil. Étant donné qu'il n'y a que deux prédicats choisis pour la phase d'enrichissement et que M=2, ces prédicats sont ajoutés comme une conjonction à chacune des réécritures candidates (Exemple 4.14).

Exemple 4.14 : Résultat de l'enrichissement des réécriture candidates de la requête initiale Q_1

Q'_{1+}(idV) :-

```
PROMOVACANCES(idV, prix, lieuDep, lieuArr, nbJours, date, heure, typeSejour,
NC, N, idT), date = '10/11/2007', lieuArr = 'Madrid', nbJours = 3,
heure>22h00, typeSejour='circuit'.
```

∪

```
LYONVACANCES(idV, prix, lieuDep, lieuArr, nbJours, date, heure, typeSejour,
NC, N, idT, idH), date = '10/11/2007', lieuArr = 'Madrid', nbJours = 3,
heure>22h00, typeSejour='circuit'.
```

Nous avons présenté intuitivement les avantages et les inconvénients des deux approches de reformulation de requêtes. Pour pouvoir mesurer ces avantages et inconvénients, la section suivante définit deux métriques d'évaluation et décrit les tests réalisés.

4. Premières évaluations des deux approches

Cette section définit deux métriques que nous utilisons pour comparer les approches de reformulation de requêtes et présente les résultats des premiers tests effectuées sur un échantillon de requêtes et de profils.

4.1 Définition des métriques

Nous souhaitons mesurer l'apport de chaque approche en considérant la proportion de prédicats du profil utilisés pendant la phase d'enrichissement de la requête et l'utilité réelle de ces prédicats. Le premier point nous conduit à définir une notion de *couverture* et le second une notion *d'utilité*.

Ces deux métriques sont habituellement utilisées pour mesurer les résultats restitués par l'exécution des requêtes et les comparer par rapport à un référentiel idéal constitué des résultats attendus par les utilisateurs. Dans notre cas, nous nous plaçons en amont de l'exécution et proposons des mesures qui tiennent compte uniquement de la sémantique des requêtes (expressions logiques des requêtes). Ces deux mesures sont évaluées par rapport au profil de l'utilisateur (prédicats) et non pas par rapport aux résultats de l'évaluation de la requête reformulée (tuples).

4.1.1 Métrique de couverture

Étant donnée une approche de reformulation de requête, la *couverture* a pour objectif de mesurer la proportion de prédicats du profil que cette approche est capable de prendre en compte pour l'enrichissement de la requête. Dans ce calcul, nous tenons compte uniquement des prédicats de sélection du profil utilisateur parce que les prédicats de jointure servent uniquement à lier, en cas de besoin, la requête aux prédicats de sélection du profil. Étant donné que les prédicats n'ont pas la même importance pour l'utilisateur, il est nécessaire de considérer leur poids dans le calcul de la couverture.

Nous considérons que les poids des prédicats du profil utilisateur sont plus une estimation des préférences de l'utilisateur faite au moment de la construction du profil qu'une préférence exacte de l'utilisateur. Qu'il soit donné manuellement par l'utilisateur ou dérivé de façon automatique par un algorithme d'analyse du feedback, le poids de chaque prédicat du profil dépend de la démarche utilisée. Pour gommer les éventuelles imperfections des valeurs des poids des prédicats du profil, nous proposons de partitionner le profil en un ensemble de groupes tels que les prédicats de chaque groupe aient relativement la même importance pour l'utilisateur et chaque prédicat n'appartiennent qu'à un seul groupe.

Soit GR le résultat du partitionnement d'un profil utilisateur P_u en N groupes :

$GR = \{GR_1, \ldots, GR_N\}$ t.q. $GR_1 \cup \ldots \cup GR_N = P_u \wedge \forall i,j \in 1..N, i \neq j, GR_i \cap GR_j = \varnothing$.

Le partitionnement du profil utilisateur peut être fait soit par l'utilisateur, soit par un algorithme de data mining. Dans ce document nous faisons l'hypothèse que le profil utilisateur est partitionné. L'étude et la recherche de la meilleure méthode de partitionnement fait partie des perspectives de travail.

Les groupes de prédicats n'ont pas la même importance pour l'utilisateur. L'importance d'un groupe GR_i, notée I_i, est fonction des poids des prédicats le composant (il est plus important de satisfaire les prédicats de plus grand poids) ainsi que du nombre de ces prédicats (il est plus intéressant de couvrir 5 prédicats sur 10 plutôt que 1 sur 2). Soient IN_i et IW_i les importances relatives d'un groupe de prédicats GR_i par rapport aux autres groupes, calculées respectivement en fonction du nombre et des poids des prédicats de GR_i. Dans notre évaluation, nous avons choisi comme fonction de calcul des IN_i le pourcentage de prédicats dans chaque groupe par rapport au nombre total de prédicats dans le profil utilisateur et comme fonction de calcul des IW_i la moyenne des poids w_j des prédicats de chaque groupe (en normalisant l'ensemble des moyennes obtenues) :

$$IN_i = \frac{|GR_i|}{|Pu|} \quad et \quad IW_i = \frac{avg(GR_i)}{\sum_{j=i}^{N} avg(GR_j)} \quad avec \quad avg(GR_i) = \frac{\sum_{p \in GR_i} w(p)}{|GR_i|},$$

où w(p) est le poids du prédicat p.

En utilisant ces deux paramètres, on peut définir le poids I_i d'un groupe GR_i comme étant la somme pondérée de IN_i et IW_i :

$$I_i = \frac{\alpha \times IN_i + \beta \times IW_i}{\alpha + \beta},$$

où α et β sont des constantes exprimant l'importance relative de IN_i par rapport à IW_i. Par exemple si le poids des prédicats dans chaque groupe est deux fois plus important que leur nombre, alors $\alpha=1$ et $\beta=2$. Ces paramètres sont des préférences de l'utilisateur. Ils peuvent être fournis explicitement par l'utilisateur ou dérivés à partir de son feedback.

Soit le partitionnement suivant du profil utilisateur P_1 (Exemple 4.2) : GR = { {e, f}, {g, h, i, j, k, l}, {m, n, o, p} } (Exemple 4.15).

Exemple 4.15 : Partitionnement du profil P_1

Profil P1 : {	VOYAGE.nbJours > 7	1.0	(e)	**GR_1**
	VOYAGE.lieuDep = 'Toulouse'	0,8	(f)	
	TRANSPORT.moyen='avion'	0.7	(g)	**GR_2**
	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)	**GR_3**
	VOYAGE.lieuDep = 'Paris'	0.3	(n)	
	HOTEL.region = 'centre ville'	0.2	(o)	
	TRANSPORT.moyen = 'car'	0.1	(p)	

Considérons que le nombre et le poids des prédicats de chaque groupe ont la même importance i.e. $\alpha=1$ et $\beta=1$. Le groupe GR1 contient 2 prédicats sur un total de 12. Son importance par rapport au nombre de prédicats est $IN_1 = 2/12 \approx 0.17$. De même, le calcul des importances IN_i des autres groupes donne : $IN_2 = 6/12 = 0.5$ et $IN_3 = 1-(0.17+0.5) = 0.33$. Pour pouvoir calculer l'importance de chaque groupe selon les poids des prédicats le composant, il faut d'abord calculer les moyennes des poids des prédicats de chaque groupe :

$avg(GR_1) = (1.0+0.8)/2 = 0.9$

$avg(GR_2) = (0.7+0.65+0.6+0.55+0.5+0.5)/6 \approx 0.58$

$avg(GR_3) = (0.35+0.3+0.2+0.1)/4 \approx 0.24$

A partir de ces moyennes, on peut évaluer les IW_i :

$IW_1 = 0.9/(0.9+0.58+0.24) \approx 0.47$

$IW_2 = 0.58/(0.9+0.58+0.24) \approx 0.34$

$IW_3 = 1-(0.47+0.34) = 0.19$

Finalement, l'importance de chaque groupe est évaluée comme la moyenne ($\alpha=\beta=1$) des importances du nombre et des poids des prédicats le composant :

$I_1 = (0.17+0.47)/2 = 0.32$

$I_2 = (0.5+0.34)/2 = 0.42$

$I_3 = (0.33+0.19)/2 = 0.26$

Notons par $cov(H_1, H_2)$, où H_1 et H_2 sont deux ensembles de prédicats, la couverture des prédicats de H_1 par les prédicats de H_2 :

$$\text{cov}(H_1, H_2) = \frac{|H_1 \cap H_2|}{|H_1|}$$

Soit *$UP(P_u, Q_u, F)$* l'ensemble de *prédicats utilisables* du profil P_u par une approche de reformulation *F* pour enrichir une requête Q_u.

Définition 4.8 : Soit $UP(P_u, Q_u, F)$ l'ensemble de prédicats utilisables du profil P_u par une approche de reformulation F pour enrichir une requête Q_u. Un prédicat p est *utilisable* par une approche de reformulation F ($p \in UP(P_u, Q_u, F)$) si p peut être utilisé dans la phase d'enrichissement de F.

Pour l'approche R(E) par exemple $UP(P_1, Q_1, R(E))$ correspond à l'ensemble de prédicats de P_1 qui ne sont pas conflictuels avec Q_1 et qui peuvent être liés à Q_1 :

$UP(P_1, Q_1, R(E)) = \{ f, g, h, i, j, k, l, m, n, o, p \}$

Pour l'approche E(R), $UP(P_1, Q_1, E(R))$ inclut uniquement les prédicats de P_1 qui sont inclus ou exprimables sur au moins une réécriture candidate de Q_1 :

$UP(P_1, Q_1, E(R)) = \{ h, j, l, n \}$

Définition 4.9 : Soit GR={GR_1, …, GR_N} la partition du profil utilisateur P_u. La *couverture pondérée*, notée par weightedCoverage(P_u, Q_u, F), de P_u par un ensemble de prédicats utilisables UP(P_u, Q_u, F) d'une approche de reformulation F, est égale à la somme pondérée des couvertures des groupes de GR par UP(P_u, Q_u, F) :

$$weightedCoverage(P_u,Q_u,\mathrm{F}) = \sum_{i=1}^{N} I_i * \mathrm{cov}(GR_i,UP(P_u,Q_u,\mathrm{F}))$$

Le calcul des couvertures pondérées des prédicats du profil P_1 dans notre exemple donne les valeurs suivantes :

weightedCoverage(P_1, Q_1, R(E)) = 0.32*cov({e, f}, { f, g, h, i, j, k, l, m, n, o, p }) + 0.42*cov({g, h, i, j, k, l}, { f, g, h, i, j, k, l, m, n, o, p }) + 0.26*cov({m, n, o, p}, { f, g, h, i, j, k, l, m, n, o, p }) =

0.32*1/2 + 0.42*6/6 + 0.26*4/4 = 0.84

weightedCoverage(P_1, Q_1, E(R)) = 0.32*cov({e, f}, { h, j, l, n }) + 0.42*cov({g, h, i, j, k, l}, {h, j, l, n }) + 0.26*cov({m, n, o, p}, { h, j, l, n }) =

0.32*0/2 + 0.42*3/6 + 0.26*1/4 = 0.275

La métrique de couverture pondérée montre à quel niveau une approche de reformulation de requêtes est capable d'utiliser le profil utilisateur, mais ne donne pas d'indication de l'utilité des prédicats pris en compte. La section suivante définit une métrique qui permet de mesurer cette utilité.

4.1.2 Métrique d'utilité

Étant donnée une approche de reformulation *F*, *l'utilité* de cette approche sert à évaluer la capacité de *F* à utiliser des prédicats du profil utilisateur qui peuvent être réécrits et qui *influencent le résultat*.

Définition 4.10 : Un prédicat *p* du profil utilisateur qui peut être réécrit, *influence* le résultat d'une requête Q_u si l'une des conditions suivantes est vérifiée :

- *p* peut être exprimé et n'est pas inclus dans la définition des sources d'au moins une réécriture candidate de Q_u ou
- *p* est conflictuel avec la définition des sources d'au moins une réécriture candidate.

Dans le premier cas, *p* permet d'ajouter une restriction supplémentaire à la requête Q_u. Un tel exemple est le prédicat « DEPART.heure>22h00 » du profil P_1 (Exemple 4.2) par rapport à la requête Q_1 (Exemple 4.4). Dans le deuxième cas *p,* permet d'éliminer les réécritures

candidates dont les sources sont conflictuelles avec *p* (par exemple le prédicat « VOYAGE.lieuDep='Paris' »).

Pour pouvoir calculer l'utilité des prédicats du profil qu'une approche de reformulation utilise pendant la phase d'enrichissement, il faut définir un référentiel. Ce référentiel est constitué des prédicats *réellement utiles* d'un profil utilisateur P_u par rapport à une requête Q_u :

Définition 4.11 : Soit $RUP(P_u, Q_u)$, l'ensemble de prédicats d'un profil utilisateur P_u *réellement utiles* à l'enrichissement d'une requête Q_u. $p \in RUP(P_u, Q_u)$ s'il satisfait les conditions suivantes :

- p peut être utilisé pour enrichir Q_u,
- p peut être réécrit et
- p influence le résultat de Q_u.

Prenons par exemple le profil P_1 (Exemple 4.2) et la requête Q_1 (Exemple 4.4). Le prédicat (e) ne peut pas être utilisé pour enrichir Q_1 (il est conflictuel avec elle), (f) ne peut pas être réécrit (il est conflictuel avec les sources de toutes les réécritures candidates de Q_1) et (j) n'influence pas le résultat de Q_1 (il est satisfait par toutes ses réécritures candidates). Ces 3 prédicats ne vérifient pas les conditions de la définition des prédicats réellement utiles et par conséquent $RUP(P_1, Q_1)$ = { g, h, i, k, l, m, n, o, p }.

Le calcul de l'utilité des prédicats du profil P_u qu'une approche de reformulation *F* utilise pendant la phase d'enrichissement doit être fait par rapport à l'ensemble de prédicats que *F* va réellement utiliser et non par rapport à ceux qu'elle est capable d'utiliser.

Définition 4.12 : L'ensemble de prédicats d'un profil utilisateur P_u *potentiellement utiles* pour l'enrichissement d'une requête Q_u par une approche de reformulation *F*, noté PUP(Pu, Qu, *F*), représente les prédicats utilisables UP(P_u, Q_u, *F*) sans les prédicats inutiles que *F* est capable de détecter.

Dans notre exemple, l'approche R(E) ne peut détecter aucun prédicat inutile i.e.

$PUP(P_1,Q_1,R(E)) = UP(P_1,Q_1,R(E))$ = { f, g, h, i, j, k, l, m, n, o, p }

Par contre l'approche E(R), peut détecter que (j) est inclus dans les définitions des sources de toutes les réécritures candidates de Q_1:

$PUP(P_u, Q_u, E(R))$ = { h, l, n }

En utilisant les définitions d'ensemble de prédicats potentiellement utiles et d'ensemble de prédicats réellement utiles, l'utilité peut être définie comme suit :

Définition 4.13 (utilité): L'utilité, notée par UT(P_u, Q_u, *F*), des prédicats de P_u, utilisés par une approche de reformulation F pour l'enrichissement d'une requête Q_u est égale au pourcentage de prédicats réellement utiles parmi l'ensemble de prédicats considérés comme utiles par F:

$$UT(P_u,Q_u,F) = \frac{\left|RUP(P_u,Q_u) \cap PUP(P_u,Q_u,F)\right|}{\left|PUP(P_u,Q_u,F)\right|}$$

En appliquant cette formule à notre exemple, on obtient les valeurs suivantes pour le calcul des utilités des deux approches :

$UT(P_1, Q_1, R(E)) = (|\{ g, h, i, k, l, m, n, o, p \} \cap \{ f, g, h, i, j, k, l, m, n, o, p \}|) / |\{ f, g, h, i, j, k, l, m, n, o, p \}| = 9/11 \approx 82\%$

$UT(P_1, Q_1, E(R)) = (|\{ g, h, i, k, l, m, n, o, p \} \cap \{ h, l, n \}|) / |\{ h, l, n \}| = 3/3 = 100\%$

La couverture et l'utilité, ainsi définies, permettent d'évaluer les approches de reformulation de requêtes et de montrer de façon plus objective leurs caractéristiques, même si ces dernières peuvent être appréhendées intuitivement. La section suivante décrit et discute les résultats que nous avons obtenus en évaluant les deux approches de reformulation et conclut sur leurs avantages et inconvénients respectifs.

4.2 Premiers tests réalisés

Afin de montrer les limites des deux scénarios de reformulation de requêtes $\mathcal{R}(\mathcal{E})$ et $\mathcal{E}(\mathcal{R})$, nous les avons appliqués sur un modeste jeu de test qui est réalisé en utilisant le schéma virtuel, les définitions des sources et un échantillon de 4 profils et 10 requêtes de l'Annexe 1.

Les valeurs de couverture et d'utilité obtenues par les deux approches de reformulation sont présentées respectivement dans les Tableau 4.5 et Tableau 4.6. Les résultats confirment l'analyse que nous avions faite dans la section précédente et montrent que la couverture obtenue par l'approche $\mathcal{E}(\mathcal{R})$ est au plus égale à celle de l'approche $\mathcal{R}(\mathcal{E})$ et que dans la plupart des cas, l'approche $\mathcal{R}(\mathcal{E})$ possède une meilleure couverture que l'approche $\mathcal{E}(\mathcal{R})$. Ceci est dû principalement au fait que lorsque l'enrichissement de la requête initiale est fait en premier, tous les prédicats non contradictoires avec ceux de la requête peuvent être utilisés. Si un prédicat porte sur un attribut qui appartient à une relation non présente dans la requête, cette relation y est ajoutée à l'aide des prédicats de jointure. Cette extension de la portée de la requête est impossible dans l'approche $\mathcal{E}(\mathcal{R})$ où l'enrichissement est fait sur un schéma fixe qui est celui des relations sources choisies. Ceci implique que les préférences exprimées sur des attributs non présents dans ce schéma ne peuvent pas être utilisées.

En comparant les deux approches selon l'utilité des prédicats du profil qu'elles utilisent, nous pouvons remarquer que l'utilité de l'approche $\mathcal{E}(\mathcal{R})$ est toujours de 100%. En effet, dans ce cas l'enrichissement est fait sur les réécritures candidates de la requête initiale qui contiennent les prédicats des définitions des sources. On dispose donc de toutes les informations nécessaires pour l'identification des prédicats utiles (prédicats de la requête initiale, prédicats du profil utilisateur et prédicats des définitions des sources). En revanche, l'approche $\mathcal{R}(\mathcal{E})$ ne permet pas de prendre en compte les définitions des sources de données dans le choix des prédicats du profil pour l'enrichissement de la requête utilisateur. Il y a deux types de prédicats inutiles du profil utilisateur non détectés par l'approche $\mathcal{R}(\mathcal{E})$: (i) les prédicats qui sont satisfaits par l'ensemble des réécritures candidates et (ii) les prédicats qui sont conflictuels avec la définition d'au moins une source de chaque réécriture candidate de la requête utilisateur.

	P1		P2		P3		P4	
	$\mathcal{R}(\mathcal{E})$	$\mathcal{E}(\mathcal{R})$	$\mathcal{R}(\mathcal{E})$	$\mathcal{E}(\mathcal{R})$	$\mathcal{R}(\mathcal{E})$	$\mathcal{E}(\mathcal{R})$	$\mathcal{R}(\mathcal{E})$	$\mathcal{E}(\mathcal{R})$
Q1	0,82	0,45	0,91	0,91	0,86	0,86	0,90	0,60
Q2	1,00	0,36	1,00	0,77	1,00	0,67	1,00	0,43
Q3	1,00	0,82	0,73	0,73	0,86	0,86	1,00	1,00
Q4	0,91	0,54	0,91	0,91	0,91	0,91	0,90	0,60
Q5	1,00	0,36	0,73	0,50	0,86	0,52	1,00	0,43
Q6	0,82	0,64	0,91	0,91	0,71	0,71	1,00	0,62
Q7	0,82	0,36	0,73	0,50	0,71	0,38	1,00	0,43
Q8	0,73	0,46	0,83	0,59	0,91	0,52	0,76	0,49
Q9	1,00	0,54	1,00	0,77	1,00	0,67	0,85	0,59
Q10	0,72	0,72	0,66	0,66	0,62	0,62	0,90	0,90

Tableau 4.5 : Résultats de l'évaluation de la couverture

	P1		P2		P3		P4	
	$\mathcal{R}(\mathcal{E})$	$\mathcal{E}(\mathcal{R})$	$\mathcal{R}(\mathcal{E})$	$\mathcal{E}(\mathcal{R})$	$\mathcal{R}(\mathcal{E})$	$\mathcal{E}(\mathcal{R})$	$\mathcal{R}(\mathcal{E})$	$\mathcal{E}(\mathcal{R})$
Q1	0,88	1,00	0,88	1,00	1,00	1,00	0,90	1,00
Q2	0,89	1,00	0,90	1,00	1,00	1,00	0,91	1,00
Q3	0,89	1,00	0,88	1,00	1,00	1,00	0,91	1,00
Q4	0,88	1,00	0,88	1,00	1,00	1,00	0,89	1,00
Q5	0,89	1,00	1,00	1,00	1,00	1,00	1,00	1,00
Q6	1,00	1,00	0,86	1,00	1,00	1,00	0,90	1,00
Q7	1,00	1,00	0,86	1,00	1,00	1,00	0,90	1,00
Q8	1,00	1,00	0,89	1,00	1,00	1,00	0,89	1,00
Q9	0,88	1,00	0,89	1,00	1,00	1,00	0,89	1,00
Q10	1,00	1,00	0,75	1,00	1,00	1,00	0,88	1,00

Tableau 4.6 : Résultats de l'évaluation de l'utilité

	Approche $\mathcal{R}(\mathcal{E})$	Approche $\mathcal{E}(\mathcal{R})$
Couverture	+ tous les prédicats non conflictuels avec la requête initiale peuvent être utilisés.	- ne couvre pas les prédicats du profil qui ne peuvent pas être exprimés sur les schémas des sources des réécritures candidates de la requête.
Utilité	-utilise des prédicats du profil qui : ➢ ne peuvent pas être réécrits, ➢ n'influencent pas le résultat de la requête.	+ ne prend en compte que des prédicats utiles du profil utilisateur

Tableau 4.7 : Tableau récapitulatif de l'analyse des approches de reformulation de requêtes

Le Tableau 4.7 récapitule les avantages et les inconvénients des deux approches par rapport aux métriques de la section précédente. En résumé, l'approche $\mathcal{R}(\mathcal{E})$ permet de couvrir un plus grand nombre de prédicats du profil utilisateur. Cette approche prend en compte l'ensemble des prédicats utiles du profil utilisateur, mais peut utiliser des prédicats inutiles pour l'enrichissement. Son point critique est l'utilisation de prédicats qui ne peuvent pas être réécrits. À la différence de l'approche $\mathcal{R}(\mathcal{E})$, l'approche $\mathcal{E}(\mathcal{R})$ n'utilise que des prédicats utiles, mais ne permet pas de les couvrir dans leur totalité. Cette approche manque tout prédicat exprimé sur des attributs non présents dans le schéma des sources choisies pour la réécriture de la requête initiale et ne permet pas de prendre en compte dans la phase d'enrichissement les « prédicats filtres » qui permettent d'éliminer un certains nombre de sources contributives pour la réécriture de la requête initiale.

Les deux scénarios de reformulation de requêtes ne sont pas contradictoires, mais complémentaires. Chacun d'eux permet d'atteindre des objectifs différents. Si le but est de satisfaire autant de préférences que possible et si tous les prédicats du profil peuvent être réécrits, alors l'approche $\mathcal{R}(\mathcal{E})$ est préférable à $\mathcal{E}(\mathcal{R})$. Si au contraire, on veut un processus de reformulation performant dans lequel on veut limiter le nombre de prédicats à prendre en compte et si on veut utiliser uniquement des prédicats utiles, alors l'approche $\mathcal{E}(\mathcal{R})$ permet d'obtenir de meilleurs résultats.

5. Conclusion

Dans ce chapitre nous avons proposé une architecture de médiation personnalisée dans laquelle la reformulation d'une requête se fait en tenant compte des préférences de l'utilisateur et des définitions des sources de données. Nous avons montré que dans un tel contexte, la reformulation d'une requête doit être faite en combinant des techniques d'enrichissement et de réécriture. En utilisant ces techniques, nous avons proposé deux approches de reformulation de requêtes qui sont (i) enrichissement-réécriture et (ii) réécriture-enrichissement. Ces approches ont été analysées et comparées selon deux métriques (couverture et utilité) que nous avons définies. Cette analyse a permis de conclure qu'aucune des deux approches n'est satisfaisante par rapport aux deux métriques. Ceci motive notre proposition d'une approche de reformulation de requêtes avec des préférences qui nous détaillons dans le chapitre suivant.

Chapitre 5. Reformulation de requêtes guidée par le profil

Ce chapitre décrit notre approche avancée de reformulation de requêtes avec des profils. Elle consiste à alterner des étapes d'enrichissement et de réécriture de requêtes.

1. Introduction

Comme nous avons vu dans le chapitre précédent, les approches séquentielles de reformulation de requêtes qui combinent des techniques de réécriture et d'enrichissement ont un certain nombre d'inconvénients : (i) l'approche enrichissement-réécriture ($\mathcal{R}(\mathcal{E})$) peut enrichir la requête utilisateur avec des prédicats inutiles de son profil et (ii) l'approche réécriture-enrichissement ($\mathcal{E}(\mathcal{R})$) ne permet pas de couvrir tous les prédicats utiles du profil utilisateur. Dans ce chapitre, nous proposons une approche de reformulation de requêtes qui permet d'éviter ces inconvénients tout en préservant les avantages des deux approches séquentielles. Elle sera évaluée et comparée avec les deux approches précédentes pour mesurer les gains éventuels et les conditions de ces gains.

Les objectifs principaux de la nouvelle approche sont :

- obtenir une utilité de 100% et
- couvrir autant de prédicats utiles du profil utilisateur que possible.

Pour atteindre le premier objectif (avoir une utilité de 100%), il faut tenir compte des définitions des sources de données lors du choix des prédicats du profil utilisateur qui seront utilisés pour enrichir la requête. Par conséquent, la réécriture de la requête doit être faite avant son enrichissement comme dans l'approche $\mathcal{E}(\mathcal{R})$. Pour atteindre le second objectif (couvrir plus de prédicats utiles du profil utilisateur), nous proposons, d'une part, d'étendre la requête initiale avec les relations virtuelles sur lesquelles sont exprimés les prédicats du profil utilisateur et, d'autre part, de ne produire que les réécritures candidates qui peuvent être enrichies avec suffisamment de prédicats du profil utilisateur. Ceci est fait dans une perspective de minimiser le risque que certains prédicats utiles du profil ne puissent pas être exprimés sur les réécritures candidates de la requête.

Sur la base de ces réflexions, nous proposons un algorithme de reformulation de requêtes à base de profils. Il commence avec une phase de prétraitement qui étend la requête utilisateur en fonction du contenu de son profil. Ensuite la requête étendue est réécrite en élaguant les réécritures candidates non pertinentes par rapport au profil utilisateur. Finalement, les réécritures candidates pertinentes sont enrichies avec les prédicats du profil.

Le processus de reformulation est présenté sur la Figure 5.1. Il est composé de 4 étapes : expansion de la requête, identification des sources pertinentes, combinaison des sources pertinentes et enrichissement final. Notre contribution se situe au niveau de la première et la de troisième étapes (expansion de la requête et combinaison des sources pertinentes). La seconde étape (identification des sources pertinentes) est faite avec l'algorithme MiniCon [HP01] décrit dans le chapitre précédent. Plus précisément, nous utilisons les MiniCon Descriptors (MCDs) générés lors de la première phase de cet algorithme. Le choix de MiniCon est motivé par le fait que les MCDs qu'il produit peuvent être combinées sans avoir à tester la satisfaisabilité des prédicats de jointure de la requête.

La quatrième étape de notre approche produit l'enrichissement final des réécritures candidates. Ceci peut être fait en utilisant par exemple l'algorithme de Koutrika et Ioannidis [KI04a, KI05b] décrit dans le chapitre précédent.

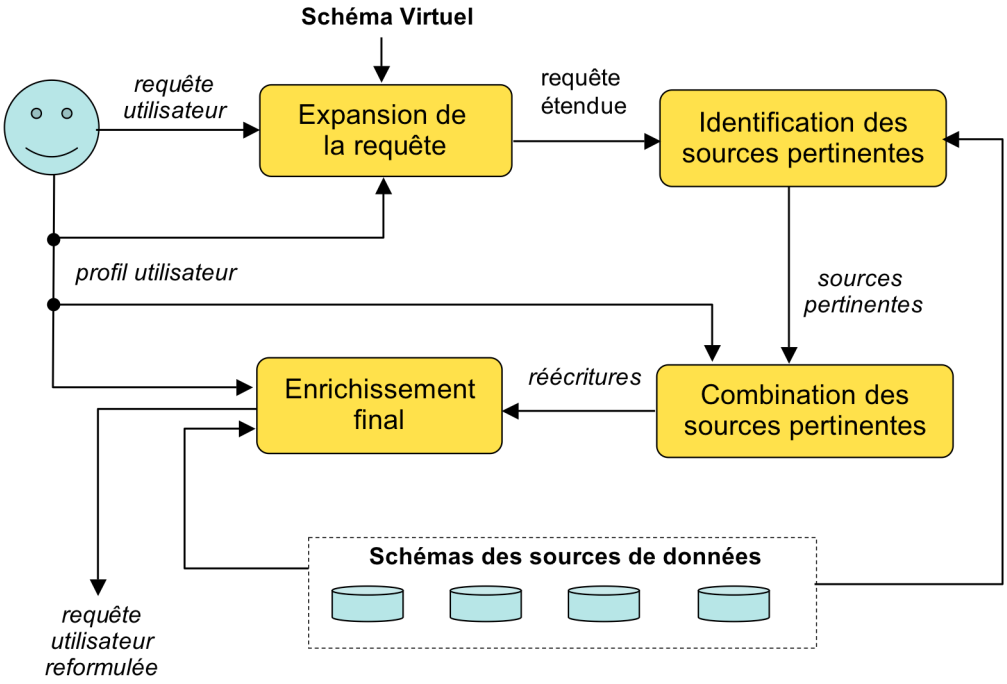


Figure 5.1 : Processus de reformulation de requêtes

Ce chapitre est organisé comme suit. La section 2 décrit le type de profil utilisateur que l'algorithme prend en compte. La section 3 détaille l'étape d'expansion de la requête. La section 4 décrit la combinaison de sources pertinentes. La section 5 conclut le chapitre.

2. Modèle de profil

Dans notre approche, nous allons utiliser un profil utilisateur qui, comparé au modèle présenté dans le chapitre 3, est restreint au domaine d'intérêt et plus précisément à un ensemble de prédicats de sélection pondérés. Ce modèle de représentation est proche de celui de Koutrika et Ioannidis [KI04a] excepté pour les prédicats de jointure qui ne sont pas inclus dans notre modèle.

Nous faisons l'hypothèse que le profil utilisateur est indépendant du schéma. L'utilisateur peut exploiter son profil sur n'importe quel schéma (réel ou virtuel) à condition d'avoir au moins un mapping partiel $\mathcal{M}$ entre le schéma et le profil. Dans ce cas, on dit que le profil utilisateur doit être *interprété* avant d'être utilisé sur un schéma particulier.

Soit P_u un profil utilisateur et S un schéma. Notons par $\mathcal{A}(Pu)$ et $\mathcal{A}(S)$ l'ensemble d'attributs de P_u et de S respectivement.

Définition 5.1 : L'*interprétation* d'un profil utilisateur P_u sur un schéma S, est le processus de matching des attributs de P_u sur les attributs de S.

Le résultat de l'interprétation est un ensemble de mappings entre des attributs qui sont sémantiquement équivalents. Ces mappings peuvent être vus comme une fonction binaire $\mathcal{M}$ entre les attributs du profil et ceux du schéma, définie comme suit :

$\mathcal{M} \subseteq \mathcal{A}(P_u) \times \mathcal{A}(S)$ et $(a, R.b) \in \mathcal{M}$ où R est une relation de S, s'il existe un mapping entre l'attribut $a \in \mathcal{A}(P_u)$ et l'attribut b de la relation R tel que $R.b \in \mathcal{A}(S)$.

Dans ce cas l'attribut *a* est dit *lié* à la relation *R* à travers l'interprétation $\mathcal{M}$, noté $\mathcal{M}\{R.b \rightarrow a\}S$. Par extension tout prédicat $p \in P_u$ exprimé sur l'attribut *a* est *lié* à la relation *R* *à travers* $\mathcal{M}$. Le problème de calcul de l'interprétation $\mathcal{M}$ d'un profil utilisateur sur un schéma donné est en dehors des problèmes traités dans cette thèse. Dans la suite, nous considérons uniquement des profils utilisateur interprétés.

Prenons par exemple un utilisateur qui a les mêmes préférences que l'utilisateur de la section 2.1 du chapitre précédent. Ces préférences peuvent être exprimées par l'ensemble de prédicats du profil P_3 de l'Exemple 5.1.

Exemple 5.1 : Exemple de profil utilisateur non interprété (P_3)

nbJours > 7	1.0	(e)
lieuDep = 'Toulouse'	0,8	(f)
moyen='avion'	0.7	(g)
heure > 22hOO	0.65	(h)
direct=VRAI	0.6	(i)
prix < 1000	0.55	(j)
nbEtoiles > 3	0.5	(k)
typeSejour = 'circuit'	0.5	(l)
niveauConfort > 2	0.35	(m)
lieuDep = 'Paris'	0.3	(n)
region = 'centre ville'	0.2	(o)
moyen = 'car'	0.1	(p)

Le profil P_4 de l'Exemple 5.2 illustre le résultat de l'interprétation du profil P_3 sur le schéma de l'Exemple 4.1. Chaque attribut de P_4 est mappé à un attribut du schéma virtuel.

Exemple 5.2 : Exemple de profil utilisateur interprété (P_4)

$\mathcal{M}$ {

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

}S_v

L'interprétation permet d'identifier les parties du profil utilisateur qui peuvent être exploitées sur un schéma particulier. Pour désigner ces parties, nous utilisons la notion de *portée* (scope) du profil utilisateur sur un schéma particulier. Elle est égale aux relations du schéma auxquelles sont liés les prédicats du profil :

Définition 5.2 : Soit S_v un schéma virtuel et soit P_u l'interprétation d'un profil utilisateur sur S_v représenté par l'ensemble de mappings $\mathcal{M}$. *La portée* de P_u sur S_v, noté $\text{SCOPE}(P_u, S_v)$, est l'ensemble de relations de S_v sur lesquelles au moins un attribut de P_u est lié :

$$\text{SCOPE}(P_u, S_v)=\{R \mid R \in S_v \wedge \exists a \in \mathcal{A}(P_u), \exists b \in \mathcal{A}(R), (a,R.b) \in \mathcal{M}\}$$

Prenons par exemple le profil interprété P_4 de l'Exemple 5.2. Sa portée inclut toutes les relations du schéma virtuel S_v (Exemple 4.1) par rapport auquel le profil a été interprété parce que P_4 contient au moins un prédicat lié à chaque relation:

SCOPE(P_4, S_v) = { VOYAGE, TRANSPORT, DEPART et HOTEL }.

L'objectif de l'interprétation d'un profil est de le rendre exploitable dans le cycle de vie des requêtes de son propriétaire. La suite de ce chapitre décrit notre approche d'exploitation du profil interprété pour la reformulation d'une requête.

3. Expansion de la requête initiale

Cette section présente la première étape de l'algorithme de reformulation de requêtes à base de profils qui est l'expansion de la requête utilisateur. Elle propose une définition du problème et décrit intuitivement la solution proposée avant de détailler le processus d'expansion.

3.1 Définition du problème

L'expansion de la requête utilisateur a pour objectif d'étendre la requête avec les relations virtuelles de la portée du profil utilisateur. Elle peut être décomposée en deux sous problèmes :

1. Identification des relations virtuelles les plus pertinentes et
2. Intégration de ces relations à la requête de l'utilisateur.

Soient S_v un schéma virtuel, P_u un profil utilisateur interprété sur S_v et Q_u une requête utilisateur exprimée sur S_v.

Identification des relations virtuelles pertinentes

Le premier sous problème consiste à trouver *la portée pertinente*, notée par RS(P_u, S_v, Q_u) (RS(P_u, S_v, Q_u) $\subseteq$ SCOPE(P_u, S_v)), qui correspond au sous-ensemble de relations de la portée du profil utilisateur qu'il est intéressant d'ajouter à Q_u. Trouver la portée pertinente revient à répondre aux questions suivantes :

- quelles relations permettent de prendre en compte d'autres prédicats du profil utilisateur que ceux pouvant être exprimés sur les relations de la requête ?
- quelles sont les relations sémantiquement liées à la requête ?
- comment choisir le sous-ensemble de relations virtuelles à ajouter dans la requête ?

La réponse à la première question, permet de déterminer si une relation de la porté du profil utilisateur est liée à d'autres prédicats de ce profil que ceux liés aux relations de la requête. L'idée principale est d'éviter de choisir dans la portée pertinente des relations qui sont déjà présentes dans la requête initiale ou qui n'ont aucun impact sur la requête si ce n'est de la rendre plus complexe.

La seconde question a pour objectif de déterminer si la sémantique du contenu des relations virtuelles est liée à la sémantique du contenu visé par la requête. Par exemple si l'utilisateur veut se renseigner sur ses metteurs en scène préférés, il est inutile de prendre en compte ses préférences sur les compagnies de production des films. Pour déterminer quelles sont les relations sémantiquement liées à la requête, il faut disposer d'une mesurer de distance sémantique. Dans notre approche nous utilisons l'intuition que des relations qui sont proches dans le graphe du schéma virtuel sont proches sémantiquement.

Finalement, la troisième question implique la définition d'une stratégie de choix des relations virtuelles. Pour pouvoir élaborer une telle stratégie il faut pouvoir comparer les relations virtuelles entre elles. L'idée principale est d'établir un ordre entre les relations

virtuelles et de choisir celles qui permettent de prendre en compte le plus de nouveaux prédicats du profil utilisateur.

Intuitivement, le choix de la portée pertinente peut être résumée par les étapes suivantes :

- sélectionner les relations virtuelles de la portée du profil utilisateur qui sont liées à des prédicats de ce profil autre que ceux liés aux relations de la requête,
- prendre en compte la distance sémantique entre les relations virtuelles sélectionnées et la requête,
- mesurer l'apport de nouveaux prédicats de chaque relation,
- choisir les relations virtuelles pour l'expansion de la requête.

Intégration des relations virtuelles pertinentes à la requête

Le second sous problème revient à ajouter les relations de la portée pertinente à la requête utilisateur. Il peut être vu comme un problème de graphes :

Soit $G_{Sv}=(V_{Sv}, E_{Sv})$ le graphe non dirigé du schéma virtuel S_v, où les nœuds V_{Sv} sont les relations de S_v et les arêtes E_{Sv} désignent les jointures entre ces relations. La requête initiale peut être représentée par le graphe non dirigé $G_{Qu}=(V_{Qu}, E_{Qu})$ dans lequel V_{Qu} et E_{Qu} représentent respectivement les ensembles des relations et des jointures de Q_u. Notons que G_{Qu} n'est pas forcément un sous-graphe de G_{Sv}. En effet, les relations virtuelles de Q_u sont un sous ensemble des relations virtuelles de S_v ($V_{Qu} \subseteq V_{Sv}$), mais la requête peut contenir des jointures autres que les jointures du schéma virtuel ($E_{Qu} \not\subset E_{Sv}$).

Dans ce contexte, ajouter une nouvelle relation virtuelle R à la requête Q_u nécessite de trouver un chemin de jointure dans G_{Sv} entre R et V_{Qu}. Ce chemin de jointure J_R peut être représenté par le graphe $G_{JR}=(V_{JR}, E_{JR})$ où V_{JR} sont les relations virtuelles du chemin et E_{JR} sont les jointures entre elles. Par conséquent, le graphe résultant de l'expansion de la requête Q_u avec la relation R est ($V_{Qu} \cup V_{JR}$, $E_{Qu} \cup E_{JR}$).

En généralisant ce procédé on peut définir le problème de l'intégration de la portée pertinente $RS(P_u, S_v, Q_u)$ à la requête initiale Q_u comme un problème de calcul du graphe résultant de l'expansion de Q_u avec toutes les relations de $RS(P_u, S_v, Q_u)$. Le résultat de cette expansion est la requête étendue Q_e, dont le graphe est :

$$G_{Qe} = (V_{Qu} \cup \bigcup_{R \in RS(Pu,Sv,Qu)} V_{J_R}, E_{Qu} \cup \bigcup_{R \in RS(Pu,Sv,Qu)} E_{J_R})$$

Le calcul de ce graphe est fait en deux étapes : (i) sélection de relations virtuelles et (ii) intégration des relations virtuelles.

3.2 Sélection des relations virtuelles

La première étape d'expansion de la requête consiste à choisir les relations virtuelles de la portée du profil utilisateur qui seront ajoutées à la requête initiale. Cette sélection est faite en tenant compte de deux paramètres :

- la longueur des chemins de jointure permettant de relier les relations virtuelles à la requête et
- la pertinence des prédicats du profil utilisateur qui sont liés aux relations virtuelles.

Le premier paramètre permet de déterminer à quel point une relation virtuelle est proche sémantiquement de la requête. L'hypothèse qui est faite ici est qu'en s'éloignant de la requête,

on s'éloigne de l'objectif visé. Par exemple si on cherche à réserver un voyage, il est plus intéressant de prendre en compte les préférences de l'utilisateur sur les hôtels des voyages plutôt que sur les restaurants des hôtels (en supposant qu'il existe une jointure entre voyage et hôtel et entre hôtel et restaurants hôteliers).

Le second paramètre du choix des relations virtuelles a pour objectif de quantifier les préférences du profil utilisateur que chaque nouvelle relation apporte à la requête. L'ajout de chaque nouvelle relation rend l'expression de la requête plus complexe et par la même occasion plus difficile à réécrire. Pour limiter ce phénomène, une relation virtuelle est ajoutée à la requête, si elle permet de prendre en compte suffisamment de préférences du profil utilisateur.

Pour tenir compte de la distance entre les relations de la requête et celles de la portée du profil utilisateur, cette distance est intégrée dans le poids des prédicats du profil. Le processus de sélection des relations virtuelles peut être décomposé en trois étapes : (i) actualisation des poids des prédicats du profil utilisateur, (ii) calcul des pertinences des relations virtuelles et (iii) choix des relations virtuelles pertinentes.

3.2.1 Actualisation des poids des prédicats du profil utilisateur

L'actualisation des poids des prédicats a pour objectif de prendre en compte la distance sémantique entre les relations virtuelles de la portée du profil utilisateur et la requête initiale. D'après [KI04a], le poids d'un prédicat du profil utilisateur décroît lorsque la distance entre la requête et la relation à laquelle il est lié augmente. Dans cet article, la fonction de calcul du nouveau poids d'un prédicat est la multiplication des poids des prédicats de jointure du profil utilisateur qui permettent de le lier à la requête. Étant donné que dans notre modèle le profil utilisateur ne contient pas de prédicats de jointure, cette formule doit être adaptée à notre contexte.

Soit *p* un prédicat du profil utilisateur et soit $R_p \in \text{SCOPE}(P_u, S_v)$ une relation à laquelle p est lié. Ajouter *p* à Q_u nécessite d'ajouter R_p à Q_u. Soit J_{Rp} le plus court chemin dans G_{Sv} entre R_p et Q_u i.e. le minimum des plus courts chemins entre R_p et les relations de Q_u. J_{Rp} est l'ensemble minimal de jointures entre une relation de Q_u et R_p. Dans certains cas, il est possible que J_{Rp} ne soit pas unique. Étant donné que nous sommes intéressé uniquement par la taille de ce chemin, l'existence de plusieurs plus courts chemins entre R_p et Q_u n'est pas importante.

Notons par $nw(p, Q_u, S_v)$ le *nouveau poids* d'un prédicat *p* par rapport à une requête Q_u et un schéma virtuel S_v :

$nw(p,Q_u,S_v) = \lambda^{|E_{J_{Rp}}|} w(p)$, où $\lambda \in [0,1]$ est une constante qui spécifie le taux de décroissance du poids des prédicats du profil et w(p) est le poids initial du prédicat *p*.

Rappelons que dans notre modèle tous les prédicats de jointure ont le même poids égale à 1. Le paramètre λ détermine la vitesse à laquelle le poids des prédicats diminue en s'éloignant de la requête initiale. Le nouveau poids d'un prédicat décroît plus rapidement lorsque la valeur de λ est petite. Lorsque λ est égale à 0, on ne prend en compte que les prédicats du profil utilisateurs liés aux relations de la requête initiale. Dans ce cas, aucune relation virtuelle n'est choisie pour étendre la requête initiale et le résultat de notre algorithme est le même que celui de l'approche $\mathcal{E}(\mathcal{R})$. Si λ est égale à 1, le poids des prédicats du profil utilisateur reste inchangé quelle que soit la taille de J_{Rp}. Il est important de remarquer qu'une grande valeur de λ a des répercussions sur les performances des étapes suivantes de l'algorithme : en ajoutant beaucoup de relations à la requête initiale, le temps de réécriture augmente de façon significative. Ce paramètre dépend du système et plus particulièrement du

schéma de médiation sur lequel l'algorithme sera appliqué. Pour des schémas avec peu de relations virtuelles qui représentent des concepts proches, la valeur de λ doit être proche de 1, tandis que pour de grands schémas virtuels (contenant beaucoup de relations) où les relations représentent des concepts éloignés, on peut prendre des petites valeurs pour λ. Le choix de la valeur de ce paramètre est une tâche de l'administrateur du système.

Prenons par exemple le profil utilisateur P_4 de l'Exemple 5.2, la requête initiale Q_1 de l'Exemple 4.4 exprimée sur le schéma virtuel S_v de l'Exemple 4.1 et considérons que $\lambda = 0.8$. Le graphe de jointure de S_v est représenté sur la Figure 5.2. Les nœuds des relations qui apparaissent dans Q_1 sont représentés par des rectangles colorés tandis que les rectangles non colorés correspondent au restant des relations de S_v qui dans cet exemple sont également des relations de la portée de P_4. Les chemins de jointure $J_{TRANSPORT}$ = {VOYAGE.idT = TRANSPORT.idT} et J_{HOTEL} = { VOYAGE.idH = HOTEL.idH } permettent de liés respectivement les relations virtuelles TRANSPORT et HOTEL à Q_1. Leur taille est la même et égale à 1. Par conséquent, le poids de tous les prédicats de P_4 liés à ces deux relations sera multiplié par $0.8^1 = 0.8$. Par exemple nw(« TRANSPORT.moyen→moyen = 'avion' », Q_1, S_v) = $0.8^1 \times 0.7 = 0.56$. Le résultat de l'actualisation des poids de tous les prédicats de P_4 est donné sur l'Exemple 5.3. Remarquons que l'ordre des prédicats du profil utilisateur peut être modifié en actualisant leurs poids comme c'est les cas dans notre exemple. Ce changement peut influencer le choix des prédicats pour la phase d'enrichissement final de la requête.

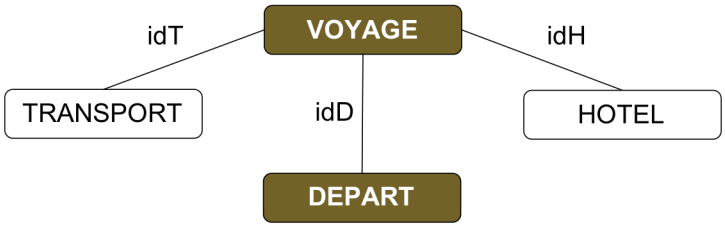


Figure 5.2 : Graphe de jointure du schéma virtuel S_v

Exemple 5.3 : Résultat de la mise à jour des poids des prédicats de P_4 en fonction de Q_1 (P_5)

$\mathcal{M}$ {	VOYAGE.nbJours→nbJours > 7	1.0	(e)
	VOYAGE.lieuDep→lieuDep = 'Toulouse'	0,8	(f)
	DEPART.heure→heure > 22hOO	0.65	(h)
	TRANSPORT.moyen→moyen = 'avion'	**0.56**	**(g)**
	VOYAGE.prix→prix < 1000	0.55	(j)
	VOYAGE.typeSejour→typeSejour = 'circuit'	0.5	(l)
	TRANSPORT.direct→direct = VRAI	**0.48**	**(i)**
	HOTEL.nbEtoiles→nbEtoiles > 3	**0.4**	**(k)**
	VOYAGE.lieuDep→lieuDep = 'Paris'	0.3	(n)
	TRANSPORT.niveauConfort→niveauConfort > 2	**0.28**	**(m)**
	HOTEL.region→region = 'centre ville'	**0.16**	**(o)**
	TRANSPORT.moyen→moyen = 'car'	**0.08**	**(p)**
} S_v			

3.2.2 Calcul des pertinences des relations virtuelles

Une fois que les poids des prédicats du profil utilisateur P_u ont été actualisés, la pertinence de chaque relation virtuelle R ∈ SCOPE(P_u, S_v), notée par *rel(R, P_u)*, est calculée.

La pertinence d'une relation virtuelle est une fonction de score qui dépend du nombre et des poids actualisés des prédicats du profil utilisateur qui sont liés à cette relation. Elle permet de définir un ordre entre les relations virtuelles en fonction de leur pertinence. Plusieurs fonctions de score peuvent être utilisées comme par exemple la somme des poids des prédicats liés à une relation, leur moyenne etc. Pour notre approche, nous utilisons comme fonction de score *la couverture pondérée* du profil utilisateur par les prédicats liés à chaque relation virtuelle (section 4.1.1 du chapitre précédent). Le choix de la couverture pondérée est motivé par le fait qu'elle intègre à la fois les deux paramètres importants qui sont le nombre et le poids des prédicats du profil utilisateur.

Prenons par exemple le partitionnement suivant du profil actualisé P_5 (Exemple 5.3) GR = { {e, f}, {g, h, i, j, k, l}, {m, n, o, p} } et la requête initiale Q_1 (Exemple 4.4). Le résultat du calcul des importances des groupes de P_5 est : I_1=0.365, I_2=0.41, I_3=0.225. La portée du profil utilisateur P_5 comprend l'ensemble des relations du schéma virtuel S_v : SCOPE(P_5, S_v) = { VOYAGE, TRANSPORT, DEPART, HOTEL }. Les ensembles de prédicats de P_5 liés à chacune des relations de sa portée sont :

VOYAGE → {e, f, j, l, m}

TRANSPORT→ {g, i, m, p}

DEPART→ {h}

HOTEL→ {k, o}

Par conséquent, leurs pertinences respectives sont :

$$rel(VOYAGE, P_5) = \sum_{i=1}^{3} I_i C(GR_i, \{e,f,j,l,m\}) = 2/2*0.365 + 2/6*0.41 + 1/4*0.225 \approx 0.558$$

$$rel(TRANSPORT, P_5) = \sum_{i=1}^{3} I_i C(GR_i, \{g,i,m,p\}) = 0/2 * 0.365 + 2/6 * 0.41 + 2/4 * 0.225 \approx 0.249$$

$$rel(DEPART, P_5) = \sum_{i=1}^{3} I_i C(GR_i, \{h\}) = 0/2 * 0.365 + 1/6 * 0.41 + 0/4 * 0.225 \approx 0.068$$

$$rel(HOTEL, P_5) = \sum_{i=1}^{3} I_i C(GR_i, \{k,o\}) = 0/2 * 0.365 + 1/6 * 0.41 + 1/4 * 0.225 \approx 0.125$$

Ce résultat permet d'ordonner les relations de SCOPE(P_5, S_v). La portée triée qui en résulte est : SCOPE(P_5, S_v) = { VOYAGE, TRANSPORT, HOTEL, DEPART }. Cet ordre, ainsi que les valeurs des pertinences obtenues vont servir à choisir les relations virtuelles qui seront ajoutées à la requête initiale.

3.2.3 Choix des relations virtuelles

La dernière étape de la sélection des relations virtuelles qui seront ajoutées à la requête initiale est le choix de la portée pertinente du profil utilisateur. Ce choix doit tenir compte non seulement des liens $\mathcal{M}$ entre le profil utilisateur et le schéma virtuel, mais également de l'expression de la requête initiale parce qu'il n'est pas nécessaire d'ajouter à la requête des relations virtuelles qui s'y trouvent déjà.

Définition 5.3 : Soit SCOPE(P_u, S_v), la portée de l'interprétation P_u d'un profil utilisateur sur un schéma virtuel S_v et soit Q_u une requête initiale exprimée sur S_v. La portée pertinente, notée RS(P_u, S_v, Q_u), de P_u par rapport à Q_u est l'ensemble de relations de SCOPE(P_u, S_v) qui n'appartiennent pas à Q_u et qui satisfont un critère de choix $\gamma(R)$:

$$RS(P_u, S_v, Q_u) = \{ R \mid R \in SCOPE(P_u, S_v) \wedge R \notin Q_u \wedge \gamma(R) \}$$

Le critère de choix $\gamma(R)$ est basé sur la pertinence des relations virtuelles de la portée du profil utilisateur. Il permet d'identifier l'ensemble de relations virtuelles ayant la meilleure pertinence qui va étendre la requête. C'est une préférence de l'utilisateur qui spécifie les conditions qu'une relation virtuelle doit remplir pour faire partie de la portée pertinente du profil utilisateur. Il peut être défini de plusieurs manières en spécifiant un seuil de pertinence ou le nombre de Top K relations. Voici quelques exemples de critères de choix :

- $|RS(P_u, S_v, Q_u)| \leq K$. Dans ce cas on choisit les K meilleures relations de la portée du profil utilisateur non présentes dans la requête. Il est important de noter que K n'est pas forcément le nombre de nouvelles relations qui seront ajoutées à la requête. S'il n'y a pas de jointure directe entre une des relations choisies et celles de la requête d'autres relations virtuelles doivent être ajoutées pour pouvoir effectuer cette jointure,
- $rel(R, P_u) \geq \sigma$. Ce critère permet de choisir les relations virtuelles qui permettent d'exprimer suffisamment de nouveaux prédicats du profil utilisateur. Si la fonction de calcul de la pertinence des relations est la couverture pondérée du profil utilisateur, ce critère identifie les relations virtuelles qui permettent de couvrir au moins σ % du profil utilisateur,
- $\sum_{R_i \in (V_{Qu} \cup RS(Pu,Sv,Qu))} rel(R_i) \geq \mu$. Dans cet exemple, le critère de choix garantit que la somme des pertinences des relations de la requête étendue sera supérieure à un seuil μ. En prenant la couverture du profil utilisateur comme fonction de calcul de la pertinence d'une relation, l'utilisateur peut spécifier quelle part de son profil doit être prise en compte au minimum pendant la reformulation de sa requête.

Dans notre approche nous utilisons le dernier critère de choix parce qu'il est plus intuitif et il est facilement exprimable par l'utilisateur.

Supposons que l'utilisateur souhaite qu'au moins 70% de son profil soit pris en compte ($\mu \geq 0.7$). Sur l'exemple de la section précédente, la somme des pertinences des relations de la requête initiale est :

rel(VOYAGE, P_5) + rel(DEPART, P_5) = 0.558+0.068 = 0.626

Les relations restantes de SCOPE(P_5, S_v) sont TRANSPORT et HOTEL. Comme transport est la relation la plus pertinente des deux, elle est choisie en premier ce qui donne :

(rel(VOYAGE, P_5) + rel(DEPART, P_5)) + rel(TRANSPORT, P_5) = 0.626 + 0.249 = 0.875

Le résultat obtenu dépasse le seuil fixé ce qui fait que la portée pertinente est composée uniquement de la relation TRANSPORT : RS(P_5, S_v, Q_1) = {TRANSPORT}.

L'intégralité de l'algorithme de sélection de la portée pertinente est présenté sur la Figure 5.3. Il prend en entrée la requête initiale, le schéma virtuel, le profil utilisateur et un

critère de choix des relations virtuelles. Premièrement, les poids des prédicats du profil utilisateur sont actualisés en fonction des distances entre les relations virtuelles auxquelles ils sont liés et les relations de la requête initiale. Ensuite, la pertinence de chaque relation de la portée du profil utilisateur est calculée. Finalement, les relations virtuelles qui satisfont le critère de choix sont retournées comme résultat. Ces relations sont utilisées dans la deuxième étape de l'expansion de la requête.

Algorithm SelectVirtualRelations

Input:

Q_u: initial query,

S_v: virtual schema,

P_u: user profile,

$\gamma(R)$: criterion for virtual relation selection

Output: RS : Relevant scope of P_u over S_v according to Q_u

Begin

$P'_u = \{p \mid p \in P_u \wedge w(p) = nw(p, Q_u, S_v)\}$

For each $R \in SCOPE(P'_u, S_v)$

Compute the relevance $rel(R, P'_u)$

Return $RS(P'_u, S_v, Q_u) = \{R \mid R \in SCOPE(P'_u, S_v) \wedge R \notin Q_u \wedge \gamma(R)\}$

End

Figure 5.3. Algorithme de sélection des relations virtuelles

3.3 Intégration des relations virtuelles

Une fois les relations virtuelles de la portée pertinente choisies, il reste à les intégrer à la requête initiale. Pour chaque relation virtuelle R qu'on veut ajouter, il peut être nécessaire d'ajouter plusieurs autres relations virtuelles permettant de joindre R à la requête. Il est important de minimiser le nombre de relations ajoutées parce que chaque nouvelle relation augmente le temps de réécriture de la requête.

La minimisation du nombre de nouvelles relations est un problème proche de celui de l'arbre de Steiner. Le problème de l'arbre de Steiner (STP) [HRW92] est défini pour un graphe non dirigé G = (V, E) où à chaque arête e ∈ E est associé un poids positif. Le STP consiste à trouver l'arbre couvrant minimal qui relie un sous-ensemble W de sommets (W ⊆ V).

Dans notre contexte, le graphe G est le graphe du schéma virtuel G_{Sv} et W représente l'union des relations de la requête initiale et celles de la portée pertinente ($W = V_{Qu} \cup RS$). Comme le profil utilisateur ne contient pas prédicats de jointure, le poids de toutes les arêtes de G_{Sv} est égal à 1. La principale différence entre STP et le problème de l'intégration des relations virtuelles est que, dans ce dernier, on doit préserver la sémantique de la requête initiale. Par conséquent, les jointures de la requête initiale forment un graphe non réductible qui doit faire partie de la solution.

Le problème de l'arbre de Steiner est connu pour être NP-complet. Par conséquent, la recherche d'une solution optimale est un processus difficile qui peut prendre beaucoup de temps. Comme la reformulation de la requête est une des étapes de son cycle de vie qui est faite en temps réel, il est difficilement envisageable de trouver des solutions optimales. Pour cette raison, nous utilisons une heuristique permettant d'ajouter la portée pertinente du profil utilisateur à sa requête. Nous avons choisi l'heuristique Minimum Cost Path Heuristic (MPH) [TM80] parce qu'elle permet de préserver les jointures de la requête initiale. MPH est un

algorithme itératif qui, à chaque étape, ajoute le minimum des plus courts chemins entre la requête et les relations virtuelles qui n'ont pas encore été ajoutées.

En utilisant MPH, il se peut qu'à une étape il y ait plusieurs plus courts chemins ayant la cardinalité minimale. Dans ce cas, le choix du chemin qui sera ajouté est fait selon la pertinence des relations qui composent chaque chemin.

Définition 5.4 : La pertinence d'un chemin de jointure JP = R_1 — R_2 — ... — R_p dans G_{Sv}, notée par rel(JP), est égale à la somme des pertinences des relations non extrémités de ce chemin :

$$rel(JP) = \sum_{i=2}^{p-1} rel(R_i)$$

La pertinence d'un chemin de jointure ne tient pas compte des relations des extrémités de ce chemin car une de ces relations est déjà dans la requête et l'autre fait partie de la portée pertinente du profil utilisateur et toutes les relations de la portée pertinente seront ajoutées à la requête. La seule question importante est de savoir quelles relations intermédiaires seront ajoutées, en plus de la portée pertinente. Étant donné que les relations virtuelles qui n'appartiennent pas à la portée du profil utilisateur ont une pertinence de zéro, seules les relations de la portée du profil utilisateur qui n'appartiennent ni à la requête initiale ni à la portée pertinente comptent dans le calcul de la pertinence des chemins candidats. En ajoutant un chemin qui a une pertinence supérieure à zéro, on ajoute à la requête des relations virtuelles de la portée du profil utilisateur qui n'ont pas été choisies dans la portée pertinente ce qui permet d'augmenter la part du profil utilisateur qui peut être prise en compte. Si lors du choix du chemin, plusieurs ont la même pertinence, un d'entre eux est choisi au hasard.

Algorithm IntegrateVirtualRelation

Input:

Q_u: initial query,

S_v: virtual schema,

P_u: user profile with actualized weights,

$RS(P_u, S_v, Q_u)$: Relevant scope of P_u over S_v according to Q_u

Output: Expanded query

Begin

While $RS(P_u, S_v, Q_u) \neq \emptyset$

Compute the shortest paths J between relations of Q_u and relations of $RS(P_u, S_v, Q_u)$

MinSP = { sp | sp ∈ J ∧ |sp| = $\min_{sp \in J}(|sp|)$ }

Select one path sp ∈ MinSP with respect to the relevance of the virtual relations

Expand Q_u with sp

Remove from $RS(P_u, S_v, Q_u)$ the selected virtual relation for which sp is constructed

endWhile

Return Q_u

End

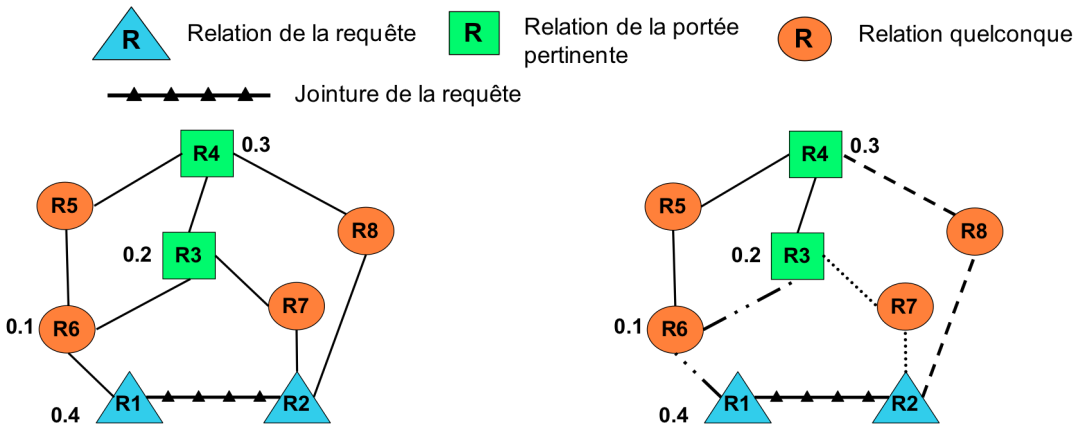
Figure 5.4. Algorithme d'intégration des relations virtuelles

La Figure 5.4 présente l'algorithme d'intégration des relations virtuelles à la requête initiale. C'est un algorithme itératif qui prend en entrée l'ensemble de relations virtuelles pertinentes RS(P_u, S_v, Q_u). À chaque itération, l'algorithme calcule les plus courts chemins dans G_{Sv} ayant les plus petites cardinalités, entre les relations de la requête et celles de RS(P_u, S_v, Q_u). Si le résultat contient plusieurs chemins, un d'entre eux est choisi en fonction des pertinences des relations qui composent les chemins. Le chemin choisi est intégré à la requête et la relation virtuelle pour laquelle il est construit est retirée de RS(P_u, S_v, Q_u). L'algorithme s'arrête lorsque toutes les relations de RS(P_u, S_v, Q_u) ont été intégrées à la requête i.e. RS(P_u, S_v, Q_u) est vide.

L'Exemple 5.4 montre le résultat de l'expansion de la requête initiale de l'Exemple 4.4, avec la portée pertinente du profil P_5 (Exemple 5.3) qui a été calculée dans la section 3.2.3 (RS(P_5, S_v, Q_1) = {TRANSPORT}). C'est un exemple très simple dans lequel il n'y a qu'une relation dans la portée pertinente et il y a une jointure directe entre elle et la relation VOYAGE de Q_1. L'expansion de Q_1 est donc triviale.

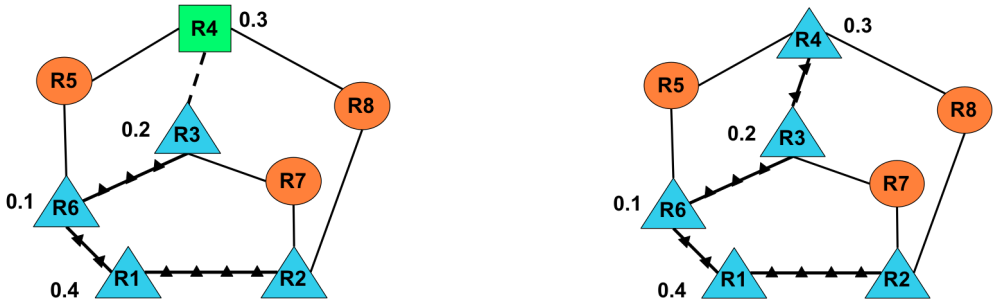
Exemple 5.4: Résultat de l'expansion de la requête initiale Q_1 (Qe)

```
Qe(idV):-
  VOYAGE(idV, prix, lieuDep, lieuArr, nbJours, typeSejour, idH, idT, idD),
  DEPART (idD, date, heure, pointRDV),
  TRANSPORT (idT, moyen, direct, niveauConfort),
    date = '10/11/2007', lieuArr = 'Madrid', nbJours = 3.
```



a) Graphe représentant les données du problème (schéma virtuel, requête initiale, relations de la portée et de la portée pertinente)

b) Plus courts chemins ayant une cardinalité minimale entre la requête et la portée pertinente



c) Requête étendue avec la première relation de RS

d) Résultat de l'expansion de la requête

Figure 5.5 : Exemple d'expansion d'une requête

Pour illustrer le fonctionnement de l'algorithme d'intégration de la portée pertinente sur un exemple plus complexe, prenons les données du graphe de la Figure 5.5a. Ce graphe contient la superposition du graphe de jointure du schéma virtuel et celui de la requête utilisateur ainsi que toutes les informations concernant la portée et la portée pertinente du profil utilisateur. Les nœuds de la requête utilisateur sont représentés par des triangles, celles de la portée pertinente par des carrés et tous les autres par des ronds. À chaque nœud correspondant à une relation de la portée du profil utilisateur est associé la pertinence de cette relation. Les lignes avec des petits triangles représentent les jointures de la requête. Dans la première itération de l'algorithme d'intégration de la portée pertinente, il y a 3 plus courts chemins de cardinalité minimale entre les relations de la portée pertinente et celle de la requête. Ces chemins sont représentés par des lignes pointillées sur la Figure 5.5b et correspondent aux chaînes R_1—R_6—R_3, R_2—R_7—R_3 et R_2—R_8—R_4. Pour décider quel chemin ajouter, il faut calculer leurs pertinences. La pertinence du premier chemin est 0.1 (rel(R_6) = 0 .1) et celle des deux autres est 0. Par conséquent, la requête est étendue avec le chemin de jointure R_1—R_6—R_3 (Figure 5.5c). Étant donné que le chemin ajouté est construit pour la relation R_3, elle est retirée de la portée pertinente. Dans la seconde itération, il reste à ajouter la relation de la portée pertinente R_4 et il y a une jointure directe entre elle et la relation R_3 de la requête (Figure 5.5c). C'est le seul chemin de cardinalité 1 et la requête est étendue avec lui (Figure 5.5d). Après cette itération, l'algorithme s'arrête car il n'y a plus de relations dans la portée pertinente. Remarquons que dans cet exemple toutes les relations de la portée du profil ont été ajoutées à la requête même si toutes ne font pas partie de la portée pertinente. Ceci est dû au fait que l'algorithme choisi en priorité les chemins contenant des relations de la portée du profil utilisateur.

La requête obtenue après la phase d'expansion est ensuite réécrite. La réécriture comprend deux phases : (i) identification des sources pertinentes et (ii) combinaison de sources pertinentes. Comme nous l'avons mentionné dans l'introduction de ce chapitre, la première phase de la réécriture est faite en utilisant la première étape de l'algorithme MiniCon qui est la construction de MCDs correspondants à des descripteurs des sources pertinentes. La section suivante présente notre algorithme qui permet de combiner ces MCDs en tenant compte du profil utilisateur.

4. Combinaison des sources pertinentes

Afin de produire des réécritures candidates de la requête utilisateur étendue, les MCDs générés pour cette requête doivent être combinés. Cette section décrit notre approche de combinaison de sources contributives en fonction du profil utilisateur.

Le principal objectif de cette étape de reformulation de la requête est de produire uniquement des réécritures candidates pertinentes i.e. celles qui peuvent être enrichies avec suffisamment de prédicats du profil utilisateur. Ceci peut être fait en écartant les combinaisons de MCDs qui ne peuvent pas produire des réécritures candidates intéressantes. Pour atteindre cet objectif il faut : (i) trouver un moyen de combiner les MCDs qui permet d'une part d'élaguer les combinaisons non pertinentes le plus tôt possible et d'autre part de limiter le nombre de combinaisons à tester et (i) définir des règles d'élagage de combinaisons de MCDs.

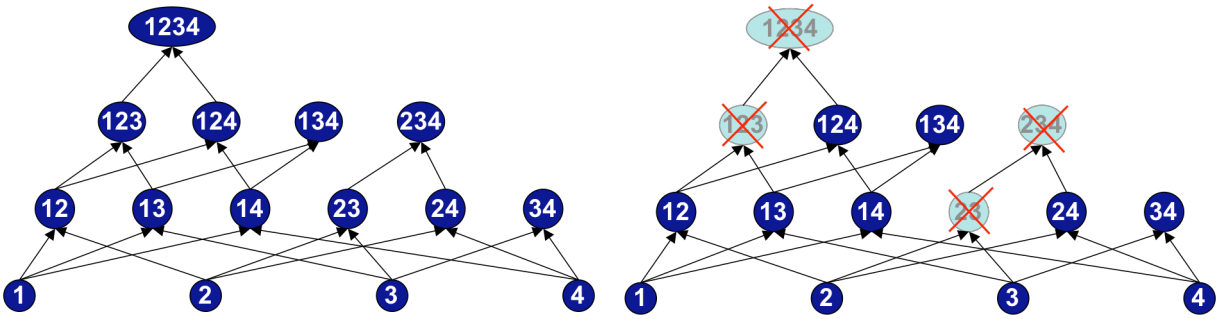
L'idée principale de notre proposition est d'explorer l'espace des combinaisons des MCDs par niveaux ce qui permet d'élaguer les combinaisons non pertinentes le plus tôt possible. Pour atteindre cet objectif, nous avons adopté l'algorithme *Apriori* [AS94].

L'algorithme *Apriori* utilise une approche itérative pour obtenir l'ensemble de « solutions » qui correspondent à des sous-ensembles d'éléments qui satisfont une propriété. C'est un algorithme par niveaux parce qu'il explore l'espace de recherche des solutions en calculant les solutions de taille i+1 à partir des solutions de taille i. Ceci est fait grâce à la fonction *AprioriGen* [AS94]. Cette fonction utilise les combinaisons qui ne diffèrent que du dernier élément (i.e. ayant les i-1 premiers éléments en commun) pour produire une nouvelle combinaison contenant les i-1 éléments en commun plus les deux éléments de la différence. Une nouvelle combinaison de i+1 éléments n'est gardée que si tous ses sous ensembles de i éléments appartiennent au niveau précédent. Si aucune combinaison n'est éliminée à aucun niveau, l'algorithme *Apriori* construit un semi treillis correspondant à toutes les combinaisons de 1, 2, 3, ..., N éléments parmi les N éléments de départ.

La Figure 5.6a montre le semi treillis construit pas Apriori lors de la combinaison de 4 éléments. En regardant le niveau 3, on peut voir que seuls les ensembles {1,2,3} et {1,2,4} ont les 2 premiers éléments en commun. Ils sont utilisés pour produire l'ensemble de 4 éléments {1,2,3,4} qui sera gardé car le niveau 3 contient tous ses sous-ensembles de 3 éléments ({1,2,3}, {1,2,4}, {1,3,4} et {2,3,4}).

L'élagage des combinaisons dans *Apriori* se fait par une propriété définie sur les ensembles d'éléments. Pour bénéficier de l'efficacité de *Apriori*, la propriété doit être *monotone* : si un ensemble ne passe pas le test, tous les sur ensembles de celui-ci ne passeront pas le test non plus.

Prenons par exemple le semi-treillis de la Figure 5.6a et supposons que l'ensemble {2,3} ne passe pas le test de la propriété. Comme le montre la Figure 5.6b, élaguer cet ensemble provoque un élagage en chaîne de tous les sur ensembles qui le contiennent. Ceci inclut aussi bien les ensembles à la construction desquels il participe (par exemple {2,3,4 }) que les autres sur ensembles le contenant (par exemple {1,2,3}).



a) Semi treillis construit par *Apriori* sans élagage des solutions

b) Propagation de l'élagage d'une solution

Figure 5.6 : Exemple de l'application de l'algorithme Apriori

Dans notre contexte, les éléments sont les MCDs et les ensembles sont des combinaisons de MCDs. Il reste à définir la propriété d'élagage des combinaisons. Une combinaison de MCDs est élaguée si au moins une des conditions suivantes est vérifiée :

- les MCDs forment une réécriture candidate,
- il y a au moins un MCD redondant avec les autres MCDs,
- les définitions des sources d'au moins 2 MCDs sont conflictuelles,
- si le score des MCDs obtenus par une fonction qui dépend de la pertinence des prédicats pouvant être exprimés sur les MCDs ne satisfait pas un seuil donné.

Remarquons que contrairement à *Apriori*, les solutions que nous recherchons font partie de la *bordure négative*. La bordure négative est formée par les plus petites combinaisons d'éléments élaguées.

Les deux premières conditions vérifient qu'une combinaison de MCDs est une réécriture et qu'elle ne contient pas de redondances. En effet, lorsqu'il y a une redondance dans un ensemble de MCDs, cette redondance est préservée dans tous les sur ensembles de celui-ci. Lorsqu'une combinaison forme une récriture candidate, tous les sous-buts de la requête sont couverts et chaque nouveau MCD sera redondant (tous ses sur ensembles seront redondants).

La troisième condition permet d'éliminer les combinaisons contenant des prédicats conflictuels qui sont exprimés sur des attributs mappés au même attribut de la requête. Comme nous traitons des requêtes conjonctives, si deux MCDs sont conflictuels, tous leurs sur ensembles contiendront cette contradiction.

Pour la quatrième condition d'élagage, nous avons besoin d'une fonction de score qui croît ou décroît de façon monotone. Nous utilisons comme telle fonction la pertinence des prédicats exclus par un ensemble de MCDs qui est appelée *pénalité*. Un prédicat est exclu par un MCD si :

- il est conflictuel avec la définition de la source du MCD,
- le MCD couvre le sous-but de la requête qui correspond à la relation virtuelle à laquelle le prédicat est lié et si ce prédicat ne peut pas être exprimé sur la source du MCD.

Propriété : La pénalité est une fonction monotone croissante.

Preuve : Nous utilisons le fait que lorsqu'un nouveau MCD est ajouté à une combinaison, l'ensemble de prédicats du profil utilisateur qui sont *exclus* par la combinaison de MCDs ne peut qu'augmenter (étant donné qu'on traite des requêtes conjonctives). L'ensemble de prédicats du profil utilisateur *exclus* par une combinaison de MCDs est égale à l'union des prédicats du profil exclus par chaque MCD séparément. Comme la pertinence d'un ensemble de prédicats croit lorsqu'on y ajoute de nouveaux prédicats, la pénalité d'un ensemble de MCDs ne peut qu'augmenter si on y ajoute un nouveau MCD.

Etant donné que la pénalité est monotone comme l'exige l'algorithme Apriori, si une combinaison de MCDs n'est pas pertinente (sa pénalité dépasse un certain seuil), alors toutes les réécritures candidates contenant cette combinaison de MCDs seront également non pertinentes. Le seuil de pénalité ρ est une préférence de l'utilisateur qui exprime le fait que lorsqu'une réécriture candidate exclue trop de prédicats du profil utilisateur (possède une pénalité supérieure à ρ), cette réécriture candidate n'est pas intéressante pour l'utilisateur. Plusieurs fonctions de score peuvent servir à calculer la pénalité d'un ensemble de prédicats exclus. Par exemple la somme des poids des prédicats, le maximum des poids etc. Dans notre approche, nous avons choisi comme fonction de pénalité la couverture pondérée des prédicats du profil utilisateur par les prédicats exclus par les MCDs (section 4.1.1 du chapitre précédent). Grâce à cette fonction, l'utilisateur peut fixer la part minimum de son profil qui

doit être utilisée pour enrichir les réécritures candidates. Si par exemple il souhaite qu'au moins 60% de son profil soit pris en compte, il peut fixer le seuil de pénalité à 0.4.

Il est important de noter qu'à la fin de la phase d'intégration de la portée pertinente dans la requête utilisateur, il est possible que toutes les relations de la portée du profil utilisateur n'aient pas été ajoutée à la requête. Par conséquent, le profil utilisateur contient des prédicats qui ne sont liés à aucune relation de la requête étendue. Ces prédicats ne pourront pas être utilisés pour enrichir les réécritures candidates de la requête étendue et vont attribuer une pénalité à toutes les réécritures candidates. Afin d'éviter ce phénomène, nous introduisons la notion de *profil utilisable* qui contient uniquement les prédicats du profil utilisateur qui ne sont pas conflictuels avec la requête étendue et qui sont liés à une des relations de la requête.

L'algorithme de combinaison des MCDs est présenté sur la Figure 5.7. Il commence par calculer la pénalité de chaque MCD et élaguer ceux qui dépassent le seuil fixé. L'algorithme utilise la fonction *AprioriGen* pour générer les combinaisons de MCDs des niveaux successifs [AS94]. À chaque niveau, les règles d'élagage définies précédemment sont appliquées. L'algorithme s'arrête lorsqu'il n'y a plus de combinaisons à explorer.

Algorithm CombineMCDs

Input:
MCD: set of generated MCDs,
P_u: user profile,
penalty({mcd_i},P) : penalty function for a set of MCDs,
ρ: penalty threshold

Output: Relevant candidate rewriting CR

Begin
i=1
L_i = {m ∈ MCD | penalty(m, P_u) ≤ ρ }
While $L_i \neq \emptyset$
C_{i+1} = AprioriGen(L_i)
For each e ∈ C_{i+1}
if(e is not redundant ∧
e is not conflicting ∧
penalty(e, P_u) ≤ ρ)
if(e is a rewriting)
CR = CR ∪ {e}
else
L_{i+1} = L_{1+1} ∪ {e}
i = i+1
endWhile
Return CR
End

Figure 5.7 : Algorithme de combinaison de MCDs

Prenons par exemple la requête étendue de l'Exemple 5.4 et le profil utilisateur P_5 de l'Exemple 5.3. Les prédicats de P_5 (k) et (o) sont liés à la relation virtuelle HOTEL qui n'est pas présente dans la requête étendue et le prédicat (e) est conflictuel avec la requête. Par conséquent, ces prédicats ne font pas partie du profil

utilisable et doivent être éliminés. Le profil utilisable résultant est présenté sur l'Exemple 5.5. Les prédicats éliminés doivent également être retirés des partitions du profil utilisateur. Les groupes de la partition du profil utilisable sont : GR = { {f}, {g, h, i, j, l}, {m, n, p} }. À partir de cette partition, on peut calculer les intérêts relatifs des prédicats de chaque groupe : I_1=0.31, I_2=0.455, I_3=0.235. Fixons le seuil de pénalité à 0.4. Ce seuil permet d'éliminer toutes les réécritures candidates qui ne permettent pas de prendre en compte 40% des prédicats du profil utilisable.

Exemple 5.5 : Profil utilisable obtenu à partir de P_5 par rapport à la requête étendue Qe

$\mathcal{M}$ {

VOYAGE.lieuDep→lieuDep = 'Toulouse'	0,8	(f)
DEPART.heure→heure > 22hOO	0.65	(h)
TRANSPORT.moyen→moyen = 'avion'	0.56	(g)
VOYAGE.prix→prix < 1000	0.55	(j)
VOYAGE.typeSejour→typeSejour = 'circuit'	0.5	(l)
TRANSPORT.direct→direct = VRAI	0.48	(i)
VOYAGE.lieuDep→lieuDep = 'Paris'	0.3	(n)
TRANSPORT.niveauConfort→niveauConfort > 2	0.28	(m)
TRANSPORT.moyen→moyen = 'car'	0.08	(p)

}$\mathbf{S}_v$

Les MCDs générés pour la requête Qe par l'algorithme MiniCon sont présenté dans le Tableau 5.1. Comparés aux MCDs qui ont été générés pour la réécriture de la requête enrichie Q^2_{1+} de l'Exemple 4.13 du chapitre précédent, il y a 3 MCDs supplémentaires couvrant le sous-but TRANSPORT. Un MCD qui vient de la source SNCF et un MCDs des deux sources de voyages PROMOVACANCES et LYONVACANCES. Ces MCDs peuvent être générés parce qu'aucun prédicat de sélection du profil utilisateur (et plus particulièrement « TRANSPORT.moyen→moyen = 'avion' ») n'a été ajouté à la requête Qe contrairement à Q^2_{1+}.

$V(\overline{Y})$	φ	G	Indice du MCD
HORAIRESDEPARTS(idD, D, H, RDV)	idD→idD, date→D, heure→H, pointRDV→RDV	2	A
TRANSPORTAERIEN(idT, LD, LA, D, H, M, DIR, NC)	idT→idT, moyen→M, direct→DIR, niveauConfort→NC	3	B
SNCF(idT, LD, LA, D, H, M, DIR, NC)	idT→idT, moyen→M, direct→DIR, niveauConfort→NC	3	C
VOYAGERPARTOUT(idT, LD, LA, D, H, M, DIR, NC)	idT→idT, moyen→M, direct→DIR, niveauConfort→NC	3	D
PROMOVACANCES(idV, P, LD, LA, NBJ, D, H, TS, NC, N, idT)	idT→idT, moyen→M, direct→DIR, niveauConfort→NC	3	E
PROMOVACANCES(idV, P, LD, LA, NBJ, D, H, TS, NC, N, idT)	idV→idV, prix→P, lieuDep→LD, lieuArr→LA, nbJours→NBJ, typeSejour→TS, idH→idH, idT→idT, idD→idD, date→D, heure→H	1, 2	F
LYONVACANCES(idV, P, LD, LA, NBJ, D, H, TS, NC, N, idT, idH)	idT→idT, moyen→M, direct→DIR, niveauConfort→NC	3	G

LYONVACANCES(idV, P, LD, LA, NBJ, D, H, TS, NC, N, idT, idH)	idV→idV, prix→P, lieuDep→LD, lieuArr→LA, nbJours→NBJ, typeSejour→TS, idH→idH, idT→idT, idD→idD, date→D, heure→H	1, 2	H

Tableau 5.1 : MCDs générés pour la requête étendue Qe

La Figure 5.8 présente le résultat de l'application de l'algorithme de combinaison de sources contributives sur les MCDs du Tableau 5.1. Pour ne pas surcharger la figure, les liens entre les éléments des différents niveaux ne sont pas représentés. Dans le premier niveau, la pénalité de tous les MCDs est évaluée et seuls les MCDs qui satisfont le seuil de pénalité sont utilisés dans les combinaisons. Dans notre exemple, tous les MCDs satisfont le seuil de pénalité et par conséquent au niveau 2, toutes les combinaisons de 2 MCDs parmi 8 doivent être vérifiées. Les combinaisons redondantes comme par exemple AF et BG sont éliminées et la pénalité des combinaisons restantes est calculée. Ensuite, les combinaisons dont la pénalité dépasse le seuil (par exemple BH et CH) ainsi que les réécritures candidates (BF, DF et DH) sont élaguées. Finalement, aucune combinaison de 3 MCDs ne peut être générées par *AprioriGen* à partir des combinaisons restantes ce qui fait que l'algorithme s'arrête. Dans notre exemple, aucune combinaison de MCDs ne doit être testé au niveau 3, alors qu'en faisant toutes les combinaisons de 3 éléments parmi 8 on obtient 56 combinaisons potentielles.

Le résultat de l'algorithme dans cet exemple sont des réécritures candidates qui peuvent être enrichies avec au moins 60% des prédicats du profil utilisable.

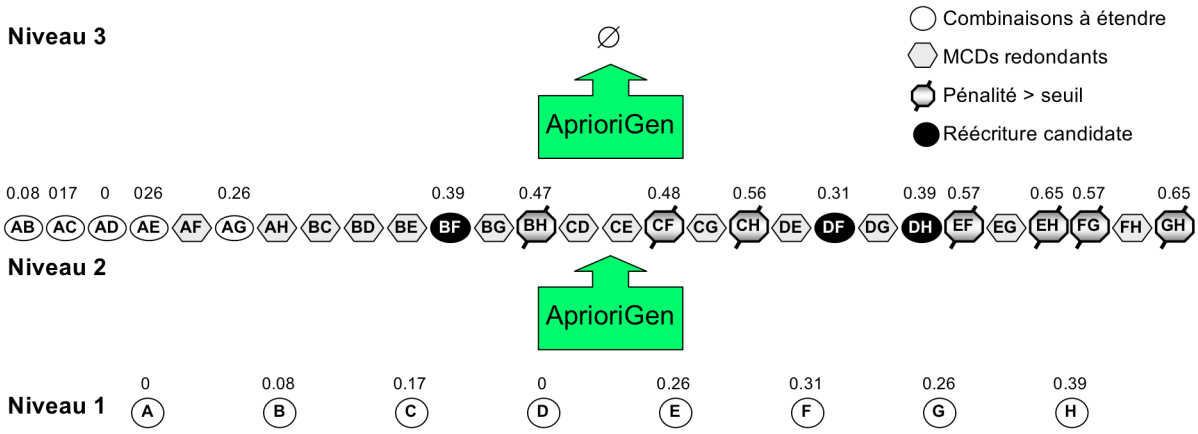


Figure 5.8 : Exemple de combinaison de MCDs

La dernière phase de la reformulation de la requête est faite en utilisant l'algorithme d'enrichissement de requêtes décrit dans la section 2.1 du chapitre 4. Le processus d'enrichissement de la requête est facilité par le fait que pendant la phase de la réécriture tous les prédicats pertinents ont été identifiés. Par conséquent, l'enrichissement peut être fait sans construire des préférences implicites et en utilisant uniquement des prédicats utiles du profil utilisateur.

5. Conclusion

Dans ce chapitre, nous avons présenté notre approche de reformulation de requêtes à base de profils. Comparée aux deux approches séquentielles, l'approche $\mathcal{R}/\mathcal{P}$ (comme l'approche $\mathcal{R}(\mathcal{E})$) est capable de prendre en compte, durant l'étape d'enrichissement de la

requête, tous les prédicats non conflictuels avec celle-ci. Ceci est possible grâce à la phase d'expansion à condition d'étendre la requête avec toutes les relations virtuelles auxquelles au moins un prédicat du profil est lié. La différence entre $\mathcal{R}(\mathcal{E})$ et $\mathcal{R}/\mathcal{P}$ est la capacité de $\mathcal{R}/\mathcal{P}$ de détecter les prédicats du profil qui ne peuvent pas être réécrits. En conclusion, notre approche possède les avantages des deux approches séquentielles, mais pas leurs inconvénients : tous et uniquement les prédicats utiles du profil utilisateur sont pris en compte pour l'enrichissement de la requête.

Chapitre 6. Évaluation des approches de reformulation de requêtes

Ce chapitre présente les évaluations des approches de reformulation de requêtes décrites dans les chapitres précédents. Il décrit le benchmark qui a servi pour faire ces évaluations ainsi que la plateforme à partir de laquelle le benchmark est dérivé. Les évaluations sont présentées à deux niveaux : niveau sémantique et niveau exécution.

1. Introduction

Nous avons défini dans cette thèse 3 approches de reformulation de requêtes dans un contexte distribué. Deux de ces approches ($\mathcal{R}(\mathcal{E})$ et $\mathcal{E}(\mathcal{R})$) sont construites à partir de la composition séquentielle d'algorithmes de réécriture et d'enrichissement tandis que la troisième approche $\mathcal{R}/\mathcal{P}$ est un algorithme de reformulation de requêtes guidé par le profil utilisateur qui alterne des étapes d'enrichissement et de réécriture. Le principal objectif de ce chapitre est d'évaluer les 3 approches de reformulation de requêtes en mettant l'accent sur $\mathcal{R}/\mathcal{P}$ qui est la plus complexe des 3. Pour atteindre cet objectif, nous avons besoin d'un benchmark qui soit définit dans un contexte distribué et qui fournit un nombre important de profils et de requêtes ainsi qu'un ensemble de résultats de référence pour chaque couple (profil, requête). Cet ensemble de référence comprend les résultats pertinents d'une requête par rapport à l'utilisateur auquel appartient le profil.

Il y a plusieurs benchmarks existants parmi lesquels on peut citer TPC [TPC07] et TREC [TREC07]. Le premier est défini pour tester la performance des serveurs de base de données tandis que le second est une plateforme de tests des algorithmes de recherche d'information. Aucun de ces benchmarks ne permet d'évaluer des approches de personnalisation. En règle générale, lorsqu'on veut valider un algorithme de personnalisation, le référentiel d'évaluation est construit avec l'implication d'utilisateurs réels. Le rôle de ces utilisateurs est de créer des profils et/ou d'identifier les résultats pertinents parmi tous les résultats d'une ou plusieurs requêtes. Les évaluations que nous présentons dans ce chapitre sont faites sur une plateforme de tests qui est à notre connaissance la première tentative de construction d'un référentiel de test pour les requêtes personnalisées.

La construction du benchmark nécessite la définition d'une plateforme de données qui soit suffisamment riche pour pouvoir en dériver toutes les informations dont nous avons besoin. Pour nos tests, nous avons utilisé une plateforme de données construite à partir de deux sources de films qui sont IMDb [IMDb07] et MovieLens [ML06]. Cette plateforme a les caractéristiques importantes suivantes :

- elle contient une grande base de données de films,
- elle propose un grand nombre de requêtes et de profils et
- elle contient un référentiel de résultats pertinents pour chaque couple (profil, requête).

A partir de cette plateforme de données, nous pouvons dériver plusieurs benchmarks de test selon les comportements que nous souhaitons observer. Chaque benchmark sera constitué

d'un ensemble de données, un ensemble de requêtes avec leurs résultats de référence et d'un ensemble de profils utilisateurs. Le choix de ces ensembles dépend des caractéristiques des algorithmes que l'on veut évaluer et des objectifs de mesure.

Ce chapitre est organisé de la manière suivante. La section 2 résume les objectifs des évaluations. La section 3 donne les grandes étapes de la construction de la plateforme de données. La section 4 résume le processus de construction du benchmark. Les sections 5 présente les résultats de l'évaluation sémantique des approches. La section 6 montre les résultats des tests d'exécution des requêtes. Finalement, la section 7 conclut le chapitre.

2. Caractéristiques des systèmes à évaluer et objectifs de l'évaluation

Les benchmarks proposés pour l'évaluation des bases de données sont généralement définis pour évaluer un type de SGBD caractérisé par son modèle (relationnel, objet, XML), son architecture (centralisé, client-serveur, distribué, …) ou ses fonctionnalités (transaction, sécurité, opérations OLAP, …). La définition des benchmarks est donc fortement influencée par ces caractéristiques. Dans notre contexte, la caractéristique principale est l'introduction de la personnalisation qui se traduit par la prise en compte du profil utilisateur durant la compilation d'une requête et l'impact de ce profil sur l'exécution des requêtes. En outre, comme nous l'avons souligné dans les chapitres précédents, nous nous plaçons dans un contexte de sources de données distribuées (architecture de médiation) où il existe un schéma virtuel et des liens sémantiques (ou requêtes de médiation) entre ce schéma virtuel et les sources de données. Les approches de personnalisations que nous avons étudiées se situent dans ce contexte, d'où l'exploitation d'algorithmes de réécriture à travers des vues.

Il est donc nécessaire que le benchmark proposé offre une simulation d'un ensemble de sources distribuées et de mécanismes de vues définissant ces sources sur le schéma virtuel.

L'évaluation d'un algorithme ou d'une approche de personnalisation a pour objectif de mesurer l'efficacité de celle-ci par rapport à des exigences fixées par l'utilisateur ou le développeur. Deux principaux facteurs de qualité nous intéressent :

- le temps de réponse des algorithmes : quel est le surcoût introduit par la personnalisation dans l'évaluation des requêtes ? Ce surcoût est-il acceptable ? Dans quelle mesure ce surcoût limite-t-il la taille ou la nature des requêtes traitées ?
- La pertinence des résultats produits : jusqu'à quel point le profil utilisateur est-il pris en compte dans la reformulation des requêtes ? Quel est l'impact de ce profil sur les résultats produits, notamment en termes de rappel et précision.

On peut également affiner les mesures en décomposant le processus en phase de compilation et phase d'exécution. En effet, dans le cadre des bases de données, il est habituel, avant d'exécuter une requête, de faire une évaluation pour sélectionner le meilleur plan d'exécution. En présence de requêtes personnalisées, cette phase prend une dimension encore plus importante dans la mesure où elle va fixer, par le processus de reformulation, la part du profil qui sera exploitée et la liste des sources de données qui seront interrogées. Nous effectuerons donc nos mesures de performance et de pertinence à deux niveaux :

- Le niveau compilation (ou niveau sémantique) : À ce niveau, nous nous intéressons, d'une part à la qualité sémantique des reformulations produites par chaque approche et, d'autre part, aux performances des algorithmes de reformulation/compilation. La qualité sémantique d'une reformulation est mesurée par la proportion de profil exploitée dans l'enrichissement des requêtes

(appelé également couverture). L'hypothèse implicite qui est faite est que la prise en compte d'une plus grande partie du profil utilisateur augmente la qualité réelle des résultats des requêtes. Cette hypothèse doit être confirmée par les mesures faites après l'exécution des requêtes. La mesure de performance va déterminer la taille des problèmes susceptibles d'être traités dans des temps de compilation acceptables pour l'utilisateur. Comme, nous l'avons vu dans les chapitres précédents, le processus de reformulation peut avoir un coût théorique exponentiel, d'où l'importance de mesurer la taille des problèmes à traiter dans des temps raisonnables.

- Le niveau exécution : Les mesures faites à ce niveau ont pour objectif de valider la justesse des reformulations faites dans la phase de compilation. Ces mesures s'expriment en termes de rappel et précision, comme c'est généralement fait dans les SRI. Le Rappel évalue la capacité d'une approche à produire tous les résultats pertinents. Il est égal à la proportion de résultats pertinents qu'une requête produit par rapport au nombre total de résultats pertinents que cette requête devrait produire. La précision évalue la capacité d'une approche de reformulation à produire uniquement des résultats pertinents. Pour la mesurer, on utilise la proportion de résultats pertinents parmi tous les résultats retournés après l'exécution d'une requête (ou un ensemble de requêtes résultantes de la reformulation). Il faut remarquer une différence principale entre les SRI et les SGBD. Dans un SRI, l'exécution d'une requête se fait en calculant la distance entre le vecteur de termes qui la représente et les vecteurs représentant les documents. L'enrichissement d'une requête peut donc produire de nouveaux résultats par rapport à ceux de la requête initiale. Dans un SGBD, l'enrichissement d'une requête revient généralement à y ajouter de nouveaux prédicats ; ce qui conduit à un filtrage plus sélectif des résultats d'une requête enrichie par rapport à la requête initiale. Pour tenir compte de ce constat, les évaluations du Rappel sont interprétées comme la perte de résultats pertinents due à la reformulation d'une requête.

3. Plateforme de tests

La première étape de la construction du benchmark des tests est la définition d'une plateforme de données suffisamment riche pour pouvoir en dériver toutes les informations nécessaires à l'évaluation des trois approches de reformulation. Pour nos besoins, nous avons défini une plateforme à partir de deux sources publiques IMDb [IMDb07] et MovieLens [ML06].

La source IMDb contient diverses informations sur une grande quantité de films (environ 859 000 au 5 Octobre 2006). Ces informations incluent la liste des acteurs, l'endroit où le film a été tourné, son genre, le type du son, la moyenne des notes (sur une échelle de 1 à 10) donnés par l'ensemble des utilisateurs qui ont évalué chaque film etc. Les données de IMDb sont réparties dans 49 fichiers. 10 de ces fichiers contiennent des données textuelles et les 39 fichiers restants contiennent des caractéristiques des films (genre, son, etc.). Le principal avantage de IMDb est le grand nombre de films qu'il contient ainsi que la richesse des descriptions de ces films. Son inconvénient est le manque du détail des évaluations des films de chaque utilisateur.

La source MovieLens est un système de recommandation qui enregistre les évaluations des utilisateurs sur les films qu'ils ont vus. Ces évaluations sont faites sur une échelle de 1 (pas pertinent) à 5 (très pertinent). Elles sont utilisées pour construire des groupes d'utilisateurs ayant les mêmes goûts cinématographiques. De cette manière, un utilisateur se

voit recommander les films que les autres utilisateurs du même groupe ont aimés. Deux ensemble de données sont disponibles pour MovieLens : un qui contient 100 000 évaluations sur 1682 films par 943 utilisateurs et un autre contenant plus d'un million d'évaluations sur environ 3900 films venant de 6040 utilisateurs. La plateforme de données est construite à partir du second ensemble.

Contrairement à IMDb, MovieLens possède des évaluations de films faites par les différents utilisateurs. Par contre, la description des films dans cette source est réduite au genre et à l'année de sortie du film. En conséquence, les deux sources sont indispensables pour la construction d'un référentiel de test pour les systèmes d'accès personnalisés. En effet, moyennant un travail de nettoyage et d'intégration, il est possible de dériver de ces deux sources non seulement une base de données volumineuse, mais aussi des profils utilisateurs et des requêtes grâce aux évaluations de films faites par les utilisateurs. À partir de ces évaluations, il est possible en effet de déduire les préférences des utilisateurs et de constituer un référentiel associant à chaque couple (profil, requête) une ensemble de résultats pertinents qui serviront de référence pour comparer les résultats produits par les requêtes personnalisées.

	IMDb	MovieLens petit	MovieLens grand
Nombre de fichiers source	49	5	3
Nombre d'utilisateurs		943	6 040
Nombre de films	858 967	1 682	3 883
Nombre d'évaluations		100 000	1 000 209

Tableau 6.1 : Statistiques du contenu des sources de données

Le Tableau 6.1 résume les statistiques du contenu des deux sources.

La construction de cette plateforme de test est faite en deux étapes : (i) acquisition des données et (ii) construction des profils et des requêtes (Figure 6.1). Chacune de ces deux étapes a nécessité la résolution de plusieurs problèmes comme l'extraction et le nettoyage des données des deux sources, le calcul de la jointure entre elles ou encore l'identification d'une méthode d'extraction de profils. Une description détaillée du processus de construction peut être trouvé dans [Per07a, Per07b]. Pour montrer l'ampleur du travail, cette section résume les principales étapes de construction de la plateforme et indique les problèmes rencontrés.

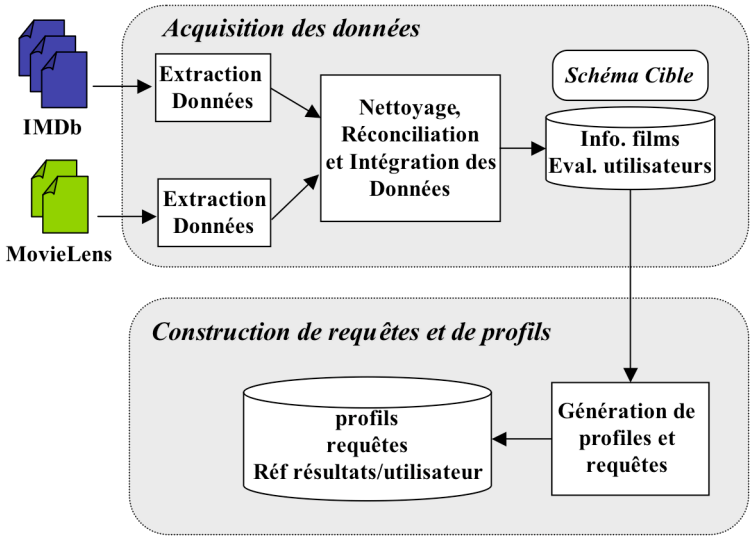


Figure 6.1 : Processus de construction de la plateforme de données

3.1 Extraction des données de IMDb et MovieLens

La première étape de la construction de la plateforme est l'extraction des données à partir des fichiers sources.

Extraction de données de MovieLens

Les données dans MovieLens sont réparties dans 3 fichiers tabulaires où les valeurs d'une ligne sont séparées par des virgules. Le traitement de fichiers structurés rend l'extraction des données relativement simple. Cependant, ces fichiers contiennent un certain nombre d'anomalies. Les problèmes rencontrés lors de l'extraction de MovieLens sont :

- la présence d'attributs multivalués,
- les valeurs nulles des attributs qui ne devrait pas l'être (ex. clés des tables),
- l'existence de doublons (tuples répliqués ou clés non uniques),
- les valeurs orphelines (tuples qui ne satisfont pas les contraintes référentielles).

Pour résoudre ces problèmes, des techniques de normalisation et de nettoyage sont utilisées [Per07a].

Les données résultantes de l'extraction de MovieLens sont stockées dans 6 tables (Figure 6.2).

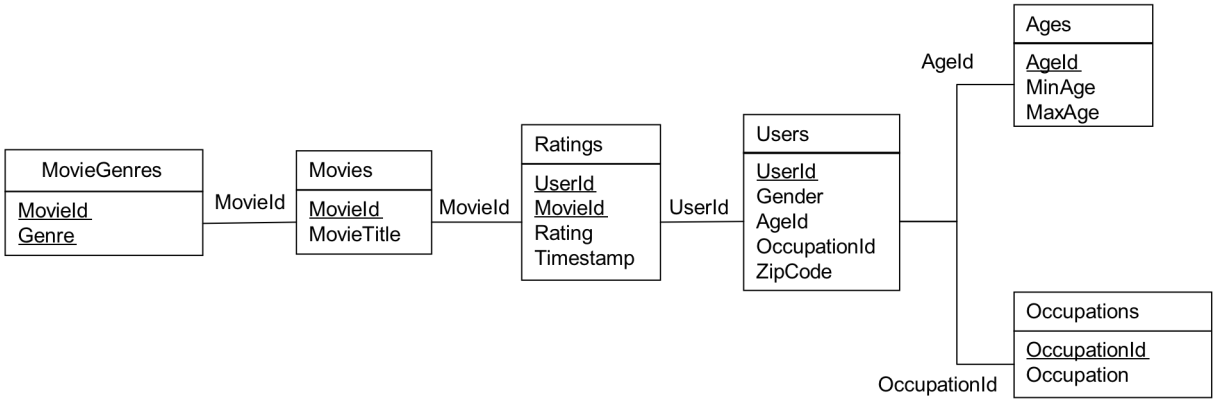


Figure 6.2 : Schéma de la BD de MovieLens

Extraction des données de IMDb

23 des 49 fichiers de IMDb sont utilisés pour construire la plateforme. Les fichiers restants ne sont pas exploités pour l'instant. L'extraction des données de IMDb est plus difficile comparée à celle de MovieLens. La première difficulté vient du fait que les fichiers textuels de IMDb ont des formats différents :

- fichiers tabulaires où chaque colonne a une taille fixe (le même nombre de caractères),
- fichiers tabulaires où les colonnes sont séparées de tabulations dont le nombre diffère souvent d'une ligne à une autre,
- texte dans lequel les valeurs sont précédées de tags,
- texte organisé hiérarchiquement.

L'extraction des données à partir des fichiers tabulaires où chaque colonne a une taille fixe ne pose pas de difficultés. Pour tous les autres formats, il est impossible d'utiliser directement des outils de chargement automatique des données dans une base de données. Par conséquent, une stratégie d'extraction particulière a du être mise en place pour chaque fichier.

En plus des problèmes liés au format des fichiers de IMDb, les données dans ces fichiers présentent des anomalies comme la présence de doublons ou d'attributs imbriqués, violation des contraintes référentielles, etc. Pour pallier ces problèmes, des techniques de nettoyage telles que l'élimination des doublons, la desimbrication des valeurs imbriquées, l'élimination des valeurs qui ne satisfont pas les contraintes référentielles, etc. sont utilisées [Per07a].

Les données extraites à partir de IMDb sont stockées dans une base de données dont le schéma est composé de 24 tables. Une partie de ce schéma est représentée dans la Figure 6.3. Le schéma complet est donné en Annexe 2.

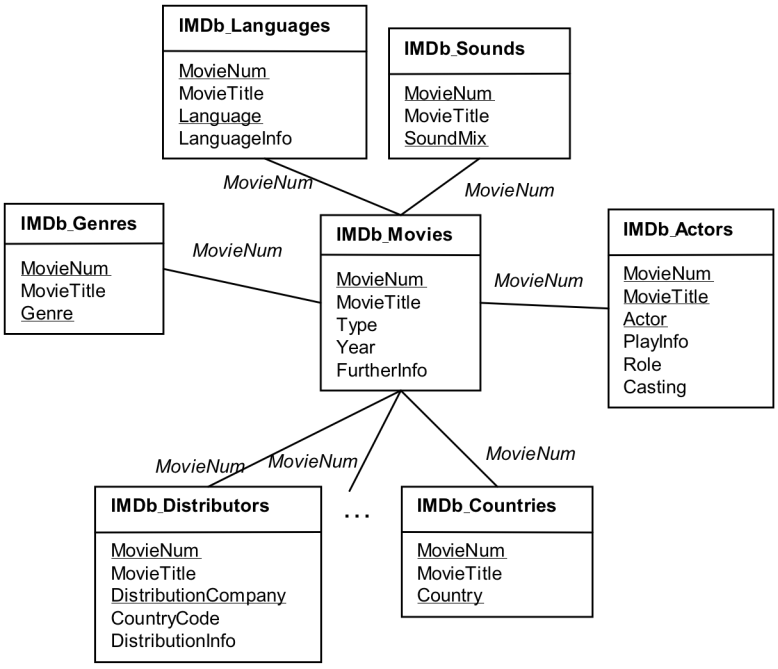


Figure 6.3 : Une partie du schéma de IMDb

Le Tableau 6.2 résume les résultats du processus d'extraction de IMDb et MovieLens. Il contient également quelques statistiques sur les relations ayant les plus grandes cardinalités de chaque base de données.

	BD IMDb	**BD MovieLens**
Taille BD	1.8 GB	60 MB
Nombre de fichiers source extraits	23	3
Nombre de tables après extraction	24	6
Nombre de films	858 967	3.883
Tables principales (nombre de tuples)	Movies (858 967) MovieActors (4 527 982) MovieLanguages (444 485)	Movies (3 883) Users (6 040) Ratings (1 000 209)

Tableau 6.2 : Statistiques sur l'extraction des données de IMDb et MovieLens

3.2 Intégration de IMDb et MovieLens

Les contenus des deux bases de données sont complémentaires. D'un coté, IMDb contient une description détaillée des films, mais n'a aucune information sur les évaluations de chaque utilisateur. De l'autre coté, MovieLens possède une grande quantité d'évaluations de films pour chaque utilisateur, mais la description des films y est très pauvre. Pour construire une plateforme de tests sur laquelle on peut évaluer des approches de personnalisation, il faut que la base de données intégrée contienne les deux ensembles

d'informations : (i) descriptions détaillées des films et (ii) évaluations de chaque utilisateur sur les films qu'il a vus. Pour cette raison, le contenu des deux sources est combiné afin d'obtenir la base de données intégrée.

Bien que les deux sources contiennent des informations sur des films, la jointure entre elles n'est pas triviale. En effet, il n'y a pas d'identifiant universel permettant de lier les films des deux sources. Le seul identifiant que IMDB et MovieLens ont en commun est le titre du film qui contient également l'année de sortie. Malheureusement, le titre d'un même film dans les deux sources n'est pas toujours le même. Ceci résulte souvent de :

- l'utilisation de caractères spéciaux (ex. ½ au lieu de 1/2),
- les erreurs de frappe,
- la transposition ou l'omission de l'article (ex. « La Bamba » et « Bamba, La »),
- la différence dans l'année de sortie du film,
- la traduction du titre dans différentes langues (ex. « Cité des enfants perdus » et « City of Lost Children »),
- l'utilisation de titres alternatifs (ex. « Sugar Hill » et « Harlem »).

Pour résoudre ces problèmes, la jointure entre IMDb et MovieLens est faite en plusieurs étapes résumées dans le Tableau 6.3. La colonne du milieu du tableau décrit brièvement la stratégie utilisée et la colonne de droite montre la progression de la quantité de films joints entre les deux sources. Presque toutes les étapes ont été accompagnées d'une vérification manuelle du résultat afin d'éliminer d'éventuels doublons. Au final, tous les films de MovieLens sont joints avec leur correspondant dans IMDb.

Étape	Description de la stratégie	Pourcentage de films joints
1	Jointure sur tous le contenu de l'identifiant du film.	79%
2	Jointure en passant par le titre des films dans le petit ensemble de MovieLens.	82%
3	Jointure en extrayant le titre du film en étranger. Certains titres dans IMDb contiennent les titre étranger entre parenthèses.	84%
4	Jointure en ignorant l'année de sortie du film.	92%
5	Jointure sur les 20 premiers caractères.	92% (34 nouvelles jointures)
6	Jointure sur les 10 premiers caractères.	94%
7	Jointure manuelle.	97%
8	Jointure manuelle par titre en utilisant les navigateurs de recherche de films que les sources proposent en ligne.	100%

Tableau 6.3 : Jointure entre IMDb et MovieLens

Le résultat de la jointure entre les deux sources est stocké dans 52 tables. Le nouveau schéma est un schéma en flocons où la relation centrale est I_Movies et les autres tables représentent les différentes caractéristiques des films. La Figure 6.4 montre une partie de ce schéma intégré, sa description complète peut être trouvée dans l'Annexe 3.

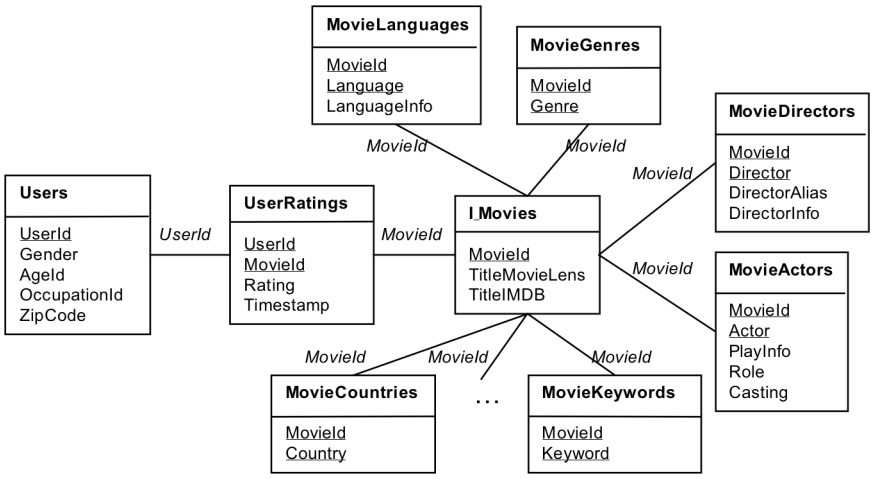


Figure 6.4 : Quelques tables du schéma intégré

	BD Intégrée
Taille BD	200 MB
Nombre de tables	52
Nombre de films	3 881
Nombre d'évaluations de films	1 000 194
Nombre d'utilisateurs	6 040

Tableau 6.4 : Contenu de la BD intégrée

Un résumé du volume de la base de donnée intégrée est présenté dans le Tableau 6.4. On y retrouve 3 881 films de MovieLens avec leur description complète venant de IMDb. La nouvelle base de données contient également toutes les évaluations de films, faites par les 6 040 utilisateurs de MovieLens.

La base de données résultante de l'intégration de IMDb et MovieLens contient toutes les informations nécessaires à l'extraction de requêtes et de profils utilisateurs qui est l'étape suivante de la construction de la plateforme.

3.3 Construction des profils et des requêtes

La seconde étape de la construction de la plateforme des tests consiste d'une part à extraire des profils et des requêtes et d'autre part à identifier les résultats pertinents pour chaque requête par rapport à chaque utilisateur.

3.3.1 Construction de profils et de requêtes

Pour générer des profils et des requêtes, un ensemble de prédicats doit être extrait à partir des données de la plateforme. Les principales étapes de la démarche d'extraction des prédicats sont:

- identifier l'ensemble de films à partir desquels sont extraits les prédicats (*ensemble de départ*),
- extraire les caractéristiques des films de l'ensemble de départ,
- pour chaque caractéristique, trouver la fréquence de son apparition dans l'ensemble de départ.

Un prédicat correspond à une caractéristique des films à laquelle est ajouté un poids. Le format d'un prédicat est :

<*table attribut opérateur valeur poids*> où opérateur ∈ {=, <, ≤, >, ≥} et poids ∈ [0, 100].

Le poids d'un prédicat correspond au pourcentage de films dans l'ensemble de départ qui le satisfont. Un poids de 100 signifie que tous les films de l'ensemble de départ possèdent la caractéristique du prédicat.

Construction des requêtes

La plateforme de données contient une requête par utilisateur. Chaque requête est construite en utilisant les prédicats extraits à partir de tous les films évalués par un utilisateur particulier. Les étapes de construction d'une requête sont :

- Tirer au hasard un nombre *i* compris entre 1 et 5. Ceci correspond au nombre de prédicats de la requête.
- Choisir au hasard *i* prédicats parmi tous les prédicats générés pour l'utilisateur.
- Intégrer les prédicats choisis à la requête « SELECT UserRatings.movieId FROM UserRatings ». Dans cette phase, chaque prédicat est ajouté comme une conjonction aux prédicats déjà présents dans la requête. En cas de besoin, la relation sur laquelle le prédicat est exprimé est également ajoutée en utilisant un ensemble de jointures prédéfinies.

Les tableaux 6.5 et 6.6 présentent les statistiques obtenues en nombre de prédicats et de relations dans les requêtes.

Nombre de prédicats	0	1	2	3	4	5
Nombre de requêtes ayant ce nombre de prédicats	1	1216	1158	1263	1194	1209

Tableau 6.5 : Nombre de prédicats dans les requêtes

Nombre de relations	1	2	3	4	5	6	7	8	9	10
Nombre de requêtes avec ce nombre de relations	1	1275	1322	1244	1007	645	364	157	23	3

Tableau 6.6 : Nombre de relations dans les requêtes

Construction des profils

Les processus d'extraction des prédicats des requêtes et des profils diffèrent principalement par les ensembles de départ qu'ils utilisent. Pour extraire les prédicats des profils, les films évalués par un utilisateur sont partitionnés en deux ensembles : un *ensemble-apprentissage* et un *ensemble-test* (Figure 6.5).

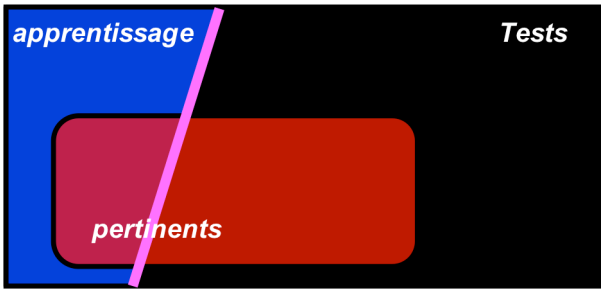


Figure 6.5 : Partitionnement des évaluations d'un utilisateur

L'ensemble-apprentissage sert à l'extraction des prédicats du profil, alors que l'ensemble-test représente les films sur lesquels les requêtes seront exécutées. Étant donné

que les prédicats d'un profil doivent capturer les préférences de l'utilisateur, seuls les films pertinents de l'ensemble-apprentissage sont utilisés comme ensemble de départ. La plateforme de données propose quatre partitionnements différents pour les films évalués par les utilisateurs. Ces partitionnement sont faits en utilisant 2 paramètres, la taille de l'ensemble-apprentissage (30% ou 50% de tous les films notés par l'utilisateur) et la note minimale d'un film pertinent (≥3 ou ≥4). Afin d'éviter les effets de bord liés au choix des films de l'ensemble-apprentissage, 5 ensembles-apprentissage sont générés de façon aléatoire pour chaque partition, ce qui donne 20 stratégies de partitionnement des films d'un utilisateur (Tableau 6.7).

Stratégie	Note min. d'un film pertinent	Taille ens.-apprentissage
1-5	3	50%
6-10	4	50%
11-15	3	30%
16-20	4	30%

Tableau 6.7 : Stratégies de partitionnement des évaluations d'un utilisateur

Pour chaque stratégie, un ensemble de prédicats est généré pour chaque utilisateur. Lors de cette phase, les prédicats ayant un poids inférieur à 10 sont écartés. En résultat de cette phase, on obtient un ensemble de prédicats pour chaque utilisateur et pour chaque stratégie différente. Au total il y a 20 * 6040 = 120 800 ensembles de prédicats qui peuvent être considérés comme des profils potentiels. Les statistiques du nombre d'utilisateurs pour lesquels un certain nombre de prédicats est généré sont représentées sur la Figure 6.6.

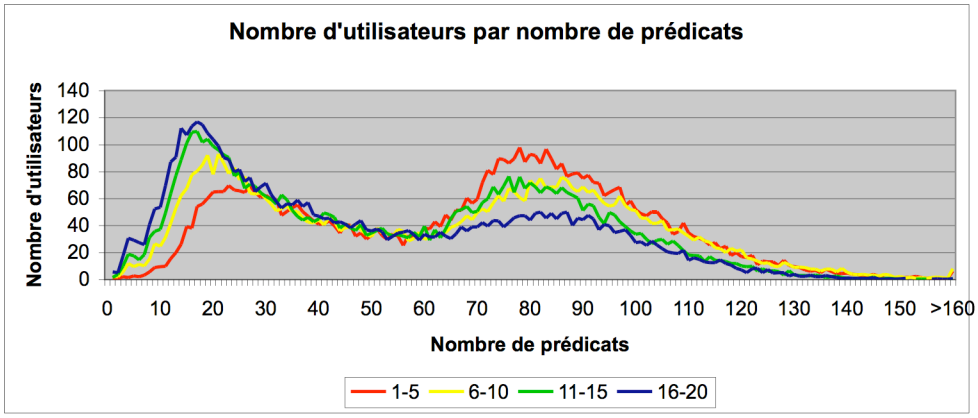


Figure 6.6 : Nombre d'utilisateurs en fonction de la taille des ensembles de prédicats

3.3.2 Référentiel de résultats pertinents

La dernière phase de la construction de la plateforme de données est l'identification des résultats pertinents pour chaque couple (profil, requête). Ce référentiel dépend des paramètres de la stratégie pour laquelle les prédicats du profil sont générés. Il est égal à l'ensemble de films pertinents retournés par l'exécution de la requête sur les films de l'ensemble-test (Figure 6.5). Un film est pertinent si sa note est supérieure ou égale à la note minimale fixée par la stratégie.

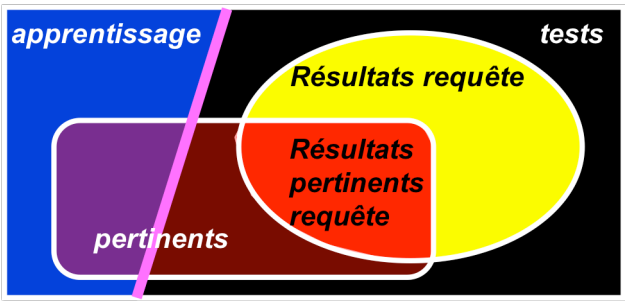


Figure 6.7 : Référentiel de résultats pertinents

La plateforme de données décrite dans cette section contient toutes les données nécessaires à l'évaluation de plusieurs approches de personnalisation. Elle propose, d'une part, une base de données dont le schéma est composé d'un grand nombre de relations et qui contient un volume important de tuples et, d'autre part, un grand nombre de profils et de requêtes avec le référentiel de résultats pertinents. Elle peut être utilisée pour définir plusieurs benchmarks selon les objectifs de test à atteindre. La section suivante décrit le benchmark que nous avons défini sur la plateforme pour évaluer les approches de reformulations de requêtes.

4. Benchmark des tests

Afin d'évaluer les approches de reformulations de requêtes sur la plateforme de données décrite dans la section précédente, il est nécessaire d'en dériver un benchmark. Les principales étapes de la construction de ce benchmark sont représentées sur la Figure 6.8.

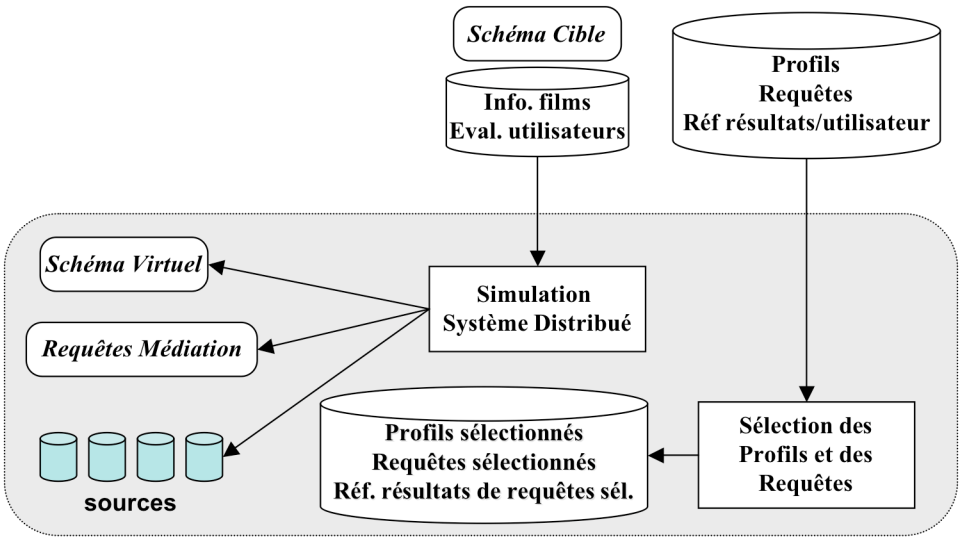


Figure 6.8 : Construction du benchmark

Le processus de construction du benchmark que nous utilisons comprend deux étapes principales :

- simulation d'un système distribué et
- choix de l'échantillon de profils et de requêtes.

La première étape consiste à construire un système distribué à partir de la plateforme. En effet, pour nos évaluations nous avons besoin d'un système distribué. Or la plateforme de données contient un schéma unique. L'objectif de cette étape est de trouver un moyen de simuler un système distribué à partir de ce qu'offre la plateforme.

La seconde étape de la construction du benchmark revient à sélectionner parmi les nombreux profils et requêtes proposés par la plateforme, un sous-ensemble qui permet de tester les approches de reformulation de requêtes.

4.1 Simulation d'un système distribué

La première étape de la construction du benchmark est la définition d'un système distribué à partir des données de la plateforme. Un système distribué a besoin de trois composants principaux : un schéma virtuel, un ensemble de sources et un ensemble de mappings (requêtes de médiation) entre les schémas source et le schéma virtuel.

Schéma virtuel

Pour notre benchmark nous avons choisi comme schéma virtuel un sous-ensemble du schéma initial de la plateforme. Les seules relations écartées sont celles qui décrivent les utilisateurs (I_Ages, I_Occupations et I_Users) parce qu'aucun prédicat des profils n'est exprimé sur ces relations. Par conséquent, elles ne sont pas utiles à l'évaluation des approches de reformulation de requêtes. Le schéma virtuel est donc composé de 49 relations (Annexe 4). Ce schéma est dit en étoile, il est constitué d'une table principale « I_Movies » et de tables secondaires. Nous verrons plus loin que cette structure aura un impact sur le coût des algorithmes de reformulation.

Sources de données et requêtes de médiation

La seconde étape de la simulation du système distribué est la construction des sources et des requêtes de médiation. Pour notre benchmark nous avons défini 52 vues sur le schéma global. Chacune de ces vues est considérée comme une source séparée. Les sources sont décrites par les requêtes qui ont servi à les construire et qui correspondent également aux requêtes de médiation.

Nombre total de vues	52
Nombre de vues contenant des prédicats de sélection	23
Nombre de vues définies sur 1 relation virtuelle	17
Nombre de vues définies sur 2 relations virtuelles	28
Nombre de vues définies sur 3 relations virtuelles	7

Tableau 6.8 : Statistiques des vues représentant les sources du système distribué

Les caractéristiques des sources sont résumées dans le Tableau 6.8. Sur les 52 sources 7 sont définies sur 3 relations virtuelles du schéma global, 28 sur 2 relations et 17 sur 1 seule relation. Les définitions de 23 sources contiennent des prédicats de sélection.

4.2 Choix des requêtes et des profils

La plateforme de données sur laquelle est construit le benchmark contient un très grand nombre de profils et de requêtes. L'exécution de toutes ces requêtes en tenant compte de tous les profils des utilisateurs correspondants nécessiterait plusieurs mois, voire plusieurs années. L'ensemble des requêtes et de profils constituant la plateforme de test n'est donc pas, à proprement parler un benchmark, mais seulement une réserve de cas permettant de générer un ou plusieurs benchmark. L'objectif de cette section est de décrire la démarche que nous avons utilisée pour sélectionner l'échantillon de requêtes et de profils sur lequel nos tests seront réalisés.

4.2.1 Choix des requêtes

Le choix de la liste des requêtes à tester est principalement guidé par le temps de réponse des algorithmes de reformulation. Ces algorithmes sont constitués de deux phases

séquentielles ou entrelacées, à savoir l'enrichissement de la requête et la réécriture de la requête.

D'après [HP01], le coût de réécriture d'une requête dépend de plusieurs paramètres : le nombre de sous buts dans les définitions des sources, le nombre de variables distinguées dans les définitions des sources et de la requête, le nombre de sources, le type de la requête (chaîne, étoile, complète, …), le nombre de sous-buts dans la requête étendue, le nombre de MCDs et de réécritures générés etc. La plupart de ces paramètres ne varient pas dans notre benchmark ; toutes les évaluations sont faites sur les mêmes définitions des sources et sur le même schéma virtuel qui a une structure en étoile. Le seul paramètre qui change est le *nombre de relations virtuelles dans la requête étendue*. C'est le paramètre qui est utilisé pour trouver la taille maximale des requêtes à reformuler. Pour atteindre cet objectif, un certain nombre de tests préliminaires ont été réalisés. Étant donné que l'approche la plus complexe que nous voulons évaluer est $\mathcal{R}/\mathcal{P}$, les tests préliminaires ont été faits en utilisant cette approche. Les résultats obtenus sont présentés sur la Figure 6.9. Les résultats de ces tests préliminaires montrent qu'au-delà de 9 relations dans la requête à réécrire, le temps de réponse dépasse largement la minute. Ajouter une relation de plus fait passer le coût à près de 20mn et ajouter 3 ou 4 relations fait passer à plusieurs dizaines d'heures de calcul. Nous poserons donc comme limite de coût de compilation la minute et comme limite à l'expansion de requête 9 relations.

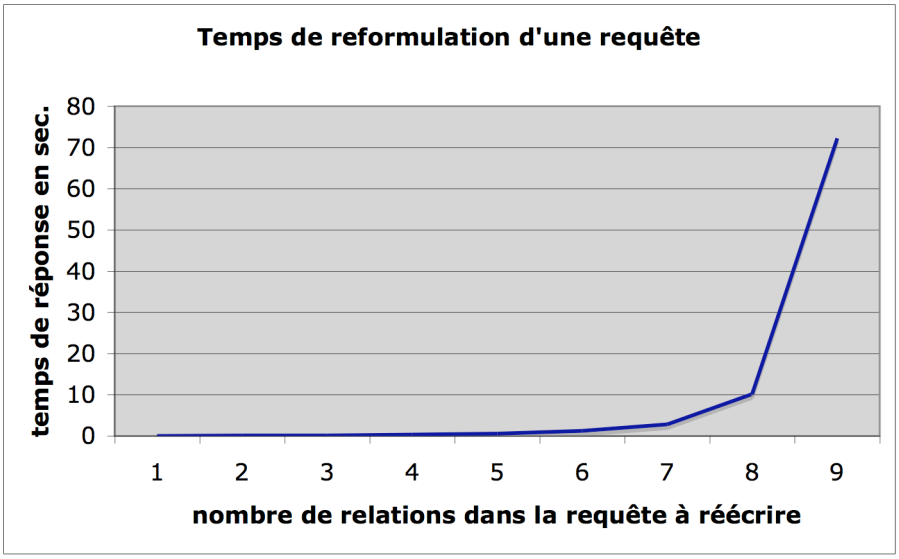


Figure 6.9 : Temps de réponse de $\mathcal{R}/\mathcal{P}$ en fonction du nombre de relations dans la requête à réécrire

Le coût de l'enrichissement dépend de l'approche utilisée. Dans $\mathcal{E}(\mathcal{R})$, il est égal au temps nécessaire d'ajouter aux réécritures candidates les prédicats du profils pouvant être exprimés sur elles. En effet, cette approche ne supporte pas l'expansion des requêtes (réécritures candidates) ce qui simplifie considérablement l'enrichissement. Dans $\mathcal{R}(\mathcal{E})$, l'enrichissement revient à : (i) parcourir le graphe du profil utilisateur afin de trouver les prédicats liés à la requête et non conflictuels avec elle, (ii) choisir les Top K prédicats du profil et (iii) intégrer les Top K prédicats à la requête. L'étape la plus coûteuse de ce processus est le parcours du graphe du profil, mais il peut être fait en temps polynomial et un seul parcours suffit pour identifier les prédicats qui sont en relation avec la requête. Dans $\mathcal{R}/\mathcal{P}$ l'enrichissement est représentée par l'expansion de la requête. Le traitement le plus coûteux de l'expansion est le calcul des plus courts chemins entre les relations de la portée pertinente du profil et les relations de la requête initiale. Ce calcul est fait autant de fois qu'il y a de relations virtuelles dans la portée pertinente. Pour nos tests nous avons fait l'hypothèse qu'ajouter les prédicats du profil à la requête revient à construire une conjonction de

prédicats. Ce traitement est trivial et son temps de calcul et négligeable devant les autres algorithmes.

En conclusion, nous pouvons retenir que le coût des algorithmes de reformulation est principalement modélisé par le coût théorique de la réécriture qui est exponentiel en nombre de relations apparaissant dans une requête. Pour les requêtes de grande taille (définies sur un grand nombre de relations), le coût de l'enrichissement reste négligeable. Pour les requêtes de petite taille, le coût de l'enrichissement peut être significatif par rapport à la réécriture ; il dépend de l'approche de reformulation utilisée.

Afin d'offrir plus de choix aux algorithmes de reformulation, les requêtes initiales ne doivent pas être définies sur beaucoup de relations virtuelles. En effet, les relations du schéma virtuel contiennent peu d'attributs ce qui implique qu'ajouter un prédicat à la requête initiale nécessite souvent d'y ajouter également la relation virtuelle sur laquelle ce prédicat est exprimé (et éventuellement d'autres relations pour permettre la jointure entre la requête et la relation). Si la requête initiale contient un grand nombre de relations virtuelles, elle laisse peu de possibilités de l'étendre or la principale différence entre les approches de reformulation vient de la phase d'expansion de la requête. Pour éviter cet effet de bord, le nombre de relations virtuelles dans la requête initiale est limité à 5. Le benchmark des tests contient des requêtes ayant entre 1 et 5 relations virtuelles. La plateforme de données contient une seule requête qui est faite sur 1 relation. Cette requête est prise dans l'échantillon du benchmark. Pour les requêtes contenant de 2 à 5 relations, 3 requêtes de chaque type sont choisies au hasard. Les caractéristiques de l'échantillon de requêtes sont résumées dans le Tableau 6.9.

Nombre de relations dans la requête	Requêtes avec 0 prédicats	Requêtes avec 1 prédicat	Requêtes avec 2 prédicats	Requêtes avec 3 prédicats	Nombre total
1	1				1
2		3			3
3		1	2		3
4		1		2	3
5			1	2	3
				Total :	13

Tableau 6.9 : Statistiques de l'échantillon de requêtes

4.2.2 Choix des profils

Le choix de l'échantillon de profils pour le benchmark est fait en 6 étapes :

- choix de la stratégie utilisée pour généré les prédicats des profils,
- choix des utilisateurs qui ont noté suffisamment de films,
- choix d'un seul prédicat par attribut et par profil,
- choix des prédicats de grand poids,
- choix d'un échantillon initial de profils,
- obtention d'un ensemble de profils de différente cardinalité pour à partir de chaque profil initial.

La première étape consiste à choisir une des 20 stratégies qui ont servi à construire les profils (voir section 3.3.1). Ce choix est fait en fonction de la note minimale d'un film pertinent dans les différentes stratégies. Rappelons qu'il y a 10 stratégies pour lesquelles un film ayant une note supérieure ou égale à 3 est considéré pertinent et 10 autres stratégies pour lesquelles cette note minimale est fixée à 4. Nos statistiques (Tableau 6.10) ont montré qu'il y

a très peu de notes de films ayant une valeur de 1 ou 2 (environs 16%). Si on veut avoir un nombre équitable de films pertinents et de films non pertinents, on doit choisir une stratégie dans laquelle un film est pertinent si sa note est supérieure ou égale à 4. Les stratégies qui satisfont ce critère sont celles de 6 à 10 et de 16 à 20 (Tableau 6.7). Pour la construction de notre benchmark nous avons utilisé les profils de la stratégie 6. En choisissant une stratégie parmi les 20 disponibles, le nombre de profils de la plateforme est réduit de 120 800 à 6 040.

Note	Nombre d'évaluations ayant cette note	Pourcentage d'évaluations ayant cette note
1	56174	5,6%
2	107556	10,8%
3	261192	26,1%
4	348965	34,9%
5	226307	22,6%

Tableau 6.10 : Statistique des notes des films de la plateforme

La seconde étape du choix des profils revient à éliminer les profils appartenant aux utilisateurs ayant noté peu de films. En effet, les requêtes produites par les approches de reformulation sont exécutées sur l'ensemble-test de l'utilisateur dont le profil a servi pour la reformulation. Pour s'assurer l'existence d'un nombre suffisant de résultats, les utilisateurs dont l'ensemble-test contient moins de 100 films sont écartés. À la fin de cette étape, il reste 747 utilisateurs qui satisfont ce critère.

La troisième étape de la construction de l'échantillon de profils représente un filtrage des prédicats de chaque profil. Ce filtrage consiste à éliminer de chaque profil tout prédicat dont le poids est inférieur au poids d'un autre prédicat exprimé sur le même attribut. Comme nous l'avons mentionné précédemment, l'enrichissement des requêtes est fait en ajoutant la conjonction des prédicats du profil. Par conséquent, pour enrichir une requête on ne peut utiliser qu'un seul prédicat par attribut. Pour tenir compte de ce constat, pour chaque profil utilisateur, un seuil prédicat par attribut est conservé (celui ayant le plus grand poids). Ce filtrage permet d'obtenir des profils ayant entre 16 et 20 prédicats (Tableau 6.11).

Nombre de prédicats	16	17	18	19	20
Nombre de profils	37	185	392	131	2

Tableau 6.11 : Taille des profils après élimination des prédicats de faible poids pour chaque attribut

La quatrième étape est un filtrage des prédicats des profils ayant des poids faibles. Le poids d'un prédicat correspond à la fréquence de son apparition dans les films pertinents de l'ensemble-apprentissage. Par conséquent, il est probable que l'ajout d'un prédicat de faible poids à la requête élimine un nombre important de résultats pertinents. Ce nombre peut être d'autant plus important que pendant l'enrichissement on ajoute souvent la conjonction de plusieurs prédicats. Pour éviter ce phénomène, il est préférable de ne prendre que des prédicats du profil utilisateur ayant des poids suffisamment grands pour ne pas filtrer beaucoup de films pertinents. Pour cette raison, les prédicats des profils ayant un poids inférieur à 80 sont éliminés. La statistique des tailles des profils après ce filtrage est représentée sur le Tableau 6.12. Le résultat de cette étape est l'obtention de 2 profils de 10 prédicats et de plusieurs profils ayant entre 2 et 9 prédicats.

Nombre de prédicats	2	3	4	4	6	7	8	9	10
Nombre de profils	19	35	31	69	339	99	103	50	2

Tableau 6.12 : Taille des profils après filtrage des prédicats ayant un poids < 80

La cinquième étape consiste à choisir un sous-ensemble de profils parmi ceux qui restent. Ce choix est dicté par l'objectif de sélectionner des profils riches (ayant beaucoup de

prédicats). Pour nos benchmark, nous avons choisi les deux profils de 10 prédicats et 3 profils de 9 prédicats pris au hasard.

Finalement, à partir des 5 profils choisi, un échantillon de 15 profils a été dérivé. Nous voulons observer les effets de la taille des profils utilisateur sur la reformulation des requêtes. Pour pouvoir le faire, chacun des 5 profils sélectionnés est utilisé pour obtenir 3 profils. Premièrement, 3 prédicats sont pris au hasard de chaque profil ce qui résulte en 5 profils de 3 prédicats. Ensuite, 3 autres prédicats (pris au hasard parmi les 6 ou 7 restants) sont ajoutés aux profils de 3 prédicats ce qui donne des profils de 6 prédicats. Le résultat de cette opération est l'obtention de 3 profils pour chaque utilisateur ayant des cardinalités différentes. On remarque que les profils de 3 prédicats sont inclus dans ceux de 6 prédicats qui de leur coté sont inclus dans les profils initiaux.

En résumé, l'échantillon de profils du benchmark est composé de 15 profils. Les étapes de la construction de cet échantillon sont résumées dans le Tableau 6.13.

Étape	Description de l'étape	Nombre de profils
	Profils de la plateforme	120 800
1	Choix de la stratégie de construction des profils (stratégie 6)	6 040
2	Filtrage des profils en fonction du nombre d'évaluations dans l'ensemble de tests (<100)	747
3	Filtrage des prédicats de faible poids	747
4	Filtrage des prédicats ayant un poids < 80	747
5	Choix des profils initiaux	5
6	Construction de profils de différente taille	15

Tableau 6.13 : Étapes de la construction de l'échantillon de profils

Il est important de noter que les poids des prédicats de la plateforme de données sont compris entre 0 et 100 or les algorithmes de reformulation de requêtes manipulent des poids entre 0 et 1. Pour pouvoir exploiter les profils de l'échantillon choisi, le poids de chaque prédicat est divisé par 100.

4.3 Récapitulatif des tests réalisés

L'objectif de cette section est de présenter l'implémentation des approches de reformulation de requêtes et de compléter la description du benchmark par le choix des paramètres des algorithmes à faire varier. Elle propose également une discussion sur les paramètres dont dépendent les différentes approches.

Implémentation des algorithmes

Pour évaluer les approches de reformulation de requêtes nous avons réalisé plusieurs implémentations. Tout d'abord, les algorithmes de réécriture et d'enrichissement que nous avons présentés dans le chapitre 4 ont été développés. Ces algorithmes ont été ensuite combinés pour obtenir les approches de reformulation séquentielles ($\mathcal{E}(\mathcal{R})$ et $\mathcal{R}(\mathcal{E})$) présentés dans le chapitre 4. Finalement, nous avons implémenté l'algorithme de réécriture à base de profils ($\mathcal{R}/\mathcal{P}$) du chapitre 5. Toutes les implémentations ont été réalisées sur la plateforme de gestion de profils décrite dans le chapitre 3. Le langage de programmation utilisé est JAVA, les données sont stockées dans une base de données Oracle 10g et les tests sont exécutés sur un ordinateur sous Windows XP ayant un processeur Pentium 4 de 3GHz et 1Go de RAM.

Paramètres à faire varier pour les tests

La reformulation de requêtes dépend de plusieurs paramètres : le taux de décroissance du poids des prédicats, le seuil de la portée pertinente et le seuil de pénalité.

Le premier de ces paramètres permet de tenir compte de la distance entre les relations de la portée du profil utilisateur et celles de la requête initiale. Étant donné que le schéma virtuel sur lequel les évaluations sont faites est un schéma en étoile, ce paramètre n'influence pas de façon significative les résultats des tests. Sa valeur est fixée à 1 pour l'ensemble des tests.

Nos tests ont été réalisés en faisant varier les valeurs des deux paramètres restants : le seuil de la portée pertinente et le seuil de pénalité. Deux séries de tests ont été réalisées en fonction de ces deux paramètres : une pour montrer le comportement des approches par rapport à la portée pertinente et une autre pour évaluer les approches selon différentes valeurs du seuil de pénalité. Pour la première série de tests, le seuil de pénalité est fixé à 0.3 et le seuil de la portée pertinente varie entre 0.1 et 0.7 avec un pas de 0.1. Le second ensemble de tests est réalisé en faisant varier le seuil de pénalité de 0.1 à 0.5 avec un pas de 0.1, pour une portée pertinente fixée à 0.5.

Schéma virtuel	49 relations
Sources	52 dont 23 contiennent des prédicats de sélection
Échantillon de requêtes	13 requêtes (1 requête avec 1 relation et 3 requêtes de chaque autre type : 2, 3, 4 et 5 relations)
Échantillon de profils	15 profils (5 avec 3 prédicats, 5 avec 6 prédicats, 3 avec 9 prédicats et 2 avec 10 prédicats)
Valeur du seuil de pénalité	De 0 à 0.5 pour portée pertinente = 0.5
Valeur de la portée pertinente	De 0.1 à 0.7 pour seuil de pénalité = 0.3

Tableau 6.14 : Résumé du contenu du benchmark

Le choix des valeurs des paramètres dont dépend la reformulation des requêtes complète la construction du benchmark. Un résumé des caractéristiques du benchmark peut être trouvé dans le Tableau 6.14.

Paramètres qui influencent les approches de reformulation de requêtes

L'évaluation des différentes approches de reformulation de requêtes passe par l'identification de l'ensemble de paramètres du benchmark qui influencent le fonctionnement de chaque approche et par l'adaptation de la sémantique de la portée pertinente au fonctionnement de cette approche.

Dans l'approche $\mathcal{E}(\mathcal{R})$ la valeur du seuil de la portée pertinente n'influence pas l'ensemble de prédicats du profils qui peuvent servir à enrichir les réécritures candidates de la requête. Dans cette approche, les réécritures candidates sont enrichies avec tous les prédicats du profil utilisateur qui peuvent être exprimés sur les schémas des sources choisies pour la réécriture. L'approche $\mathcal{E}(\mathcal{R})$ est exécutée pour chaque requête et chaque profil du benchmark.

Dans l'approche $\mathcal{R}(\mathcal{E})$, la portée pertinente est identifiée par les Top K prédicats du profil utilisateur qui sont utilisés pour l'enrichissement d'une requête. Pour lier le seuil de la portée pertinente au nombre K de prédicats, nous utilisons la formule suivante $K=\lceil \mu \times |P_u| \rceil$ où $|P_u|$ est la cardinalité du profil utilisateur et μ est la valeur du seuil de la portée pertinente. L'approche $\mathcal{R}(\mathcal{E})$ est exécutée pour chaque requête, chaque profil utilisateur et chaque valeur du seuil de la portée pertinente.

Finalement, dans l'approche $\mathcal{R}/\mathcal{P}$ le seuil de la portée pertinente sert à identifier l'ensemble minimal de relations virtuelles dont la somme des pertinences permet de satisfaire ce seuil. Les relations ainsi identifiées servent à l'expansion de la requête. Comparée à $\mathcal{R}(\mathcal{E})$, approche $\mathcal{R}/\mathcal{P}$ dépend d'un paramètre supplémentaire qui est le seuil de pénalité. Par

conséquent, $\mathcal{R}/\mathcal{P}$ est exécutée pour chaque requête, chaque profil utilisateur, chaque valeur du seuil de la portée pertinente et chaque valeur du seuil de pénalité.

Le résumé du nombre d'exécutions faites pour chaque approche sont représentés sur le Tableau 6.15. En plus des exécutions des approches de reformulation avec personnalisation, on y retrouve le nombre d'exécutions faites pour la réécriture seule des requêtes. L'algorithme de réécriture est appliqué à chaque requête et chaque utilisateur. L'exécution des réécritures candidates, produites par cet algorithme pour une requête, sur l'ensemble de tests d'un utilisateur permet de trouver les résultats pertinents pour le couple (requête, utilisateur). On remarque que dans ce cas, le référentiel de résultats pertinents est construit pour un couple (requête, utilisateur) et non pas pour une requête et un profil. En effet, le benchmark contient 3 profils pour chaque utilisateur qui sont obtenus à partir du même profil initial. Par conséquent, l'ensemble de résultats pertinents obtenus pour un utilisateur sert comme référentiel de résultats pertinents pour les 3 profils de cet utilisateur.

Approche	Paramètres influençant le nombre d'exécutions	Nombre total d'exécutions
$\mathcal{R}/\mathcal{P}$	Nombre de requêtes, nombre de profils, valeur de la portée pertinente, valeur de seuil de pénalité	2340
$\mathcal{R}(\mathcal{E})$	Nombre de requêtes, nombre de profils, valeur de la portée pertinente	1365
$\mathcal{E}(\mathcal{R})$	Nombre de requêtes, nombre de profils	195
MiniCon	Nombre de requêtes, nombre d'utilisateurs	65
	Nombre total d'exécutions	3965

Tableau 6.15 : Statistiques du nombre d'exécutions effectuées

Les exécutions des approches de reformulation sur le benchmark sont analysées afin d'évaluer la qualité de ces approches. Les résultats de cette analyse sont décrits dans les sections suivantes.

5. Évaluations au niveau sémantique

L'analyse au niveau sémantique a pour objectif de mesurer les performances des différentes approches de reformulation et la qualité sémantique des reformulations produites.

Cette section décrit les résultats obtenus par l'application des différentes approches sur le benchmark décrit dans la section précédente. Ceci est fait en montrant le comportement de ces approches en fonction des 4 paramètres utilisés pour configurer les tests : (i) nombre de prédicats dans le profil utilisateur, (ii) nombre de relations dans la requête initiale, (iii) seuil de la portée pertinente du profil utilisateur et (iv) le seuil de pénalité du calcul des réécritures candidates.

5.1 Évaluation de la couverture du profil utilisateur

Nous nous intéressons à la la couverture pondérée (weightedCoverage(P_u, Q_u, F)) comme elle a été définie dans la section 4.1.1 du chapitre 4. Pour une approche de reformulation F, un profil partitionné P_u et une requête Q_u, elle est définie comme la somme pondérée des couvertures des groupes de P_u par les prédicats de P_u que F peut utiliser pour enrichir Q_u.

Pour mesurer la couverture pondérée, nous avons fait l'hypothèse que la partition du profil utilisateur est construite de sorte à avoir une différence au plus égale à 0.05 entre le plus grand et le plus petit poids des prédicats d'un groupe. Cette hypothèse permet de partitionner

automatiquement le profil utilisateur. Le processus de partitionnement est fait par l'algorithme de la Figure 6.10. C'est un algorithme qui parcours les prédicats du profil utilisateur dans l'ordre décroissant de leurs poids. Le premier prédicat du profil sert comme point de référence pour la construction du premier groupe. Tous les prédicats suivants dont le poids n'est pas inférieur au poids du prédicat de référence moins 0.05 sont ajoutés au groupe. Le premier prédicat qui ne satisfait pas cette condition est utilisé comme prédicat de référence pour la construction du second groupe et ainsi de suite. L'algorithme s'arrête lorsque tous les prédicats du profil utilisateur sont parcourus.

Algorithm ProfilePartitioning

Input:

P_u: user profile with sorted predicates in decreasing order of their weights

Output: GR : partition of P_u

Begin

i=0

GR = ∅

while $P_u \neq \emptyset$ do

$GR_i = \{p_i / p_i \in P_u \wedge \max_{p_j \in P_u}(weight(p_j)) \in GR_i \wedge$

$(\neg \exists p_k \in P_u / p_k \notin GR_i \wedge weight(p_k) > weight(p_i)) \wedge$

$\max_{p_j \in GR_i}(weight(p_j)) - \min_{p_j \in GR_i}(weight(p_j)) \leq 0.05 \}$

$GR = GR \cup GR_i$

$P_u = P_u \setminus GR_i$

i = i+1

end do

Return GR

End

Figure 6.10 : Algorithme de partitionnement des profils utilisateur

Une fois le profil utilisateur partitionné, l'importance de chaque groupe est calculée en considérant que les poids et le nombre de prédicats dans chaque groupe ont la même importance ($\alpha=\beta=1$).

Couverture du profil en fonction du seuil de la portée pertinente

La couverture du profil utilisateur est directement liée à la valeur du seuil de la portée pertinente. Rappelons que ce seuil quantifie la part du profil que l'utilisateur souhaite exploiter dans le processus de reformulation de sa requête.

La Figure 6.11 présente les résultats de l'évaluation de la couverture du profil utilisateur obtenue par les différentes approches. Pour cette figure, la taille des profils est fixée à 6 prédicats et le nombre de relations dans les requêtes initiales est fixé à 3. Les courbes obtenues en changeant les valeurs de ces deux paramètres sont similaires.

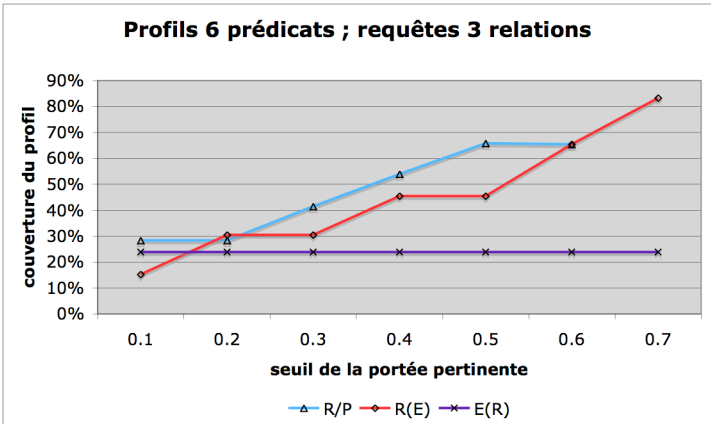


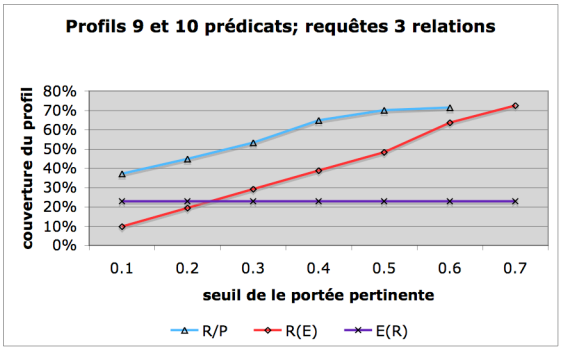
Figure 6.11 : Couverture du profil utilisateur en fonction du seuil de la portée pertinente

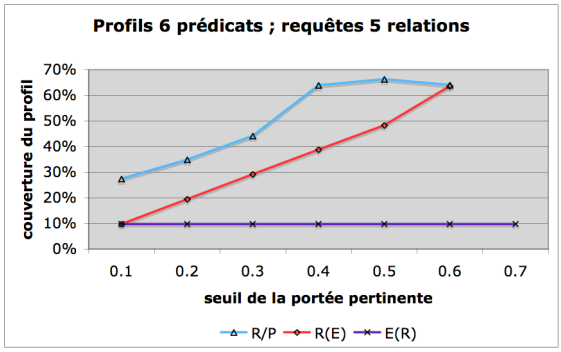
La première conclusion tirée de la Figure 6.11 est que la couverture du profil utilisateur de l'approche $\mathcal{E}(\mathcal{R})$ ne dépend pas de la valeur du seuil de la portée pertinente. En effet, les seuls prédicats du profil utilisateur que cette approche est capable de prendre en compte sont ceux liés aux relations de la requête initiale. Par conséquent, la valeur de la couverture du profil utilisateur de $\mathcal{E}(\mathcal{R})$ reste constante lorsque la valeur de la portée pertinente change.

Pour les approches $\mathcal{R}/\mathcal{P}$ et $\mathcal{R}(\mathcal{E})$, la valeur de la couverture du profil augmente lorsque la valeur de la portée pertinente augmente. C'est un résultat attendu puisque ces approches sont guidées par une prise en compte d'une partie du profil utilisateur correspondant à la valeur de la portée pertinente. Il est donc normal d'observer une telle dépendance entre la couverture du profil et la portée pertinente.

La comparaison des couvertures des profils utilisateur obtenues par $\mathcal{R}/\mathcal{P}$ et $\mathcal{R}(\mathcal{E})$ montre que $\mathcal{R}/\mathcal{P}$ permet d'obtenir une meilleure couverture que $\mathcal{R}(\mathcal{E})$ dans la plupart des cas. Cette amélioration est liée à la méthode de sélection des prédicats du profil utilisés pour enrichir les requêtes. Dans $\mathcal{R}(\mathcal{E})$ ces prédicats sont limités aux Top K prédicats du profil, alors que dans $\mathcal{R}/\mathcal{P}$ on utilise tous les prédicats pouvant être exprimés sur les sources des réécritures candidates. De plus, un des objectifs de la phase d'expansion des requêtes dans $\mathcal{R}/\mathcal{P}$ est de maximiser la part du profil utilisateur pouvant être exprimée sur les relations de la requête étendue.

Une autre observation intéressante est que le gain de couverture du profil utilisateur de $\mathcal{R}/\mathcal{P}$ par rapport à $\mathcal{R}(\mathcal{E})$ augmente avec la taille du problème qu'ils traitent. La Figure 6.12 présente les résultats des tests obtenus pour les mêmes valeurs des paramètres que dans la Figure 6.11 mais en augmentant le nombre de prédicats dans le profil utilisateur (a) ou le nombre de relations dans la requête initiale (b). Dans les deux graphiques de la Figure 6.12 l'avantage en couverture de $\mathcal{R}/\mathcal{P}$ par rapport à $\mathcal{R}(\mathcal{E})$ est plus important que celui du graphique de la Figure 6.11. Dans les deux cas, il y a plus de chemins de jointure possibles pour l'expansion de la requête initiale. Étant donné que $\mathcal{R}/\mathcal{P}$, contrairement à $\mathcal{R}(\mathcal{E})$, est conçu pour choisir les meilleurs chemins, cette approche permet d'augmenter l'exploitabilité du profil utilisateur.





a) profils de plus grande taille b) requêtes initiales avec plus de relations

Figure 6.12 : Couverture pour problèmes de plus grande taille

Couverture du profil utilisateur en fonction du nombre de prédicats dans le profil utilisateur

La Figure 6.13 illustre l'influence du nombre de prédicats du profil utilisateur sur la valeur de la couverture.

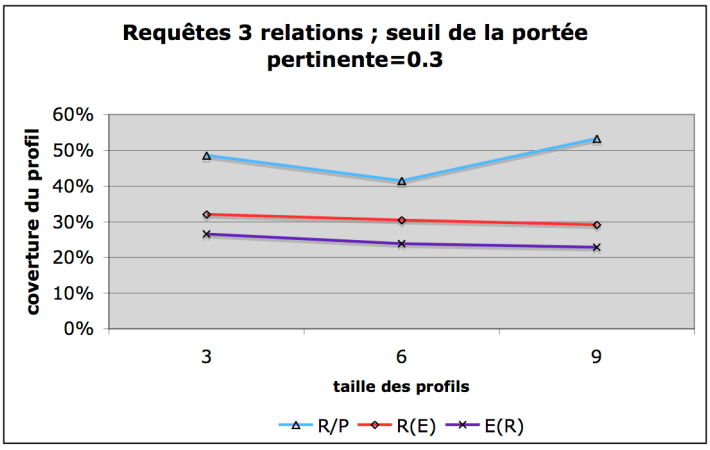
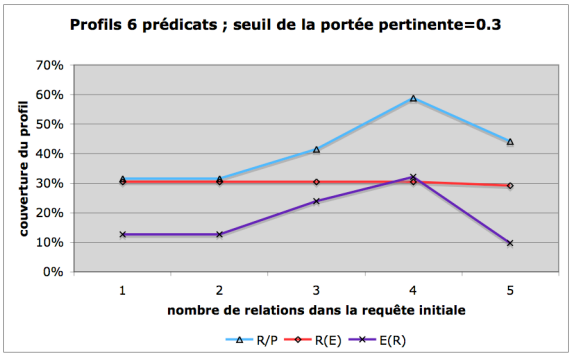


Figure 6.13 : Couverture du profil utilisateur en fonction de la taille des profils`

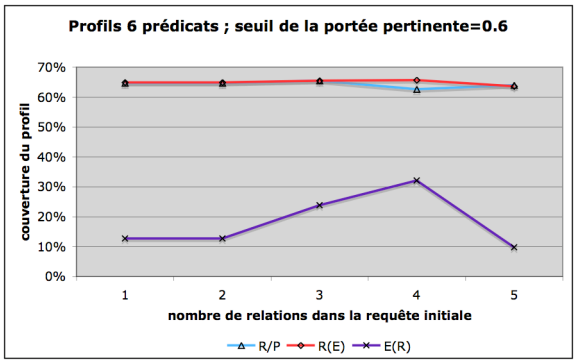
Dans notre benchmark, ce paramètre n'a pas d'influence sur la couverture du profil utilisateur. Les autres évaluations réalisées pour des requêtes différentes et pour différentes valeurs des autres paramètres nous conduisent à la même conclusion.

Couverture du profil utilisateur en fonction du nombre de relations dans la requête initiale

Comme nous l'avons mentionné précédemment, la couverture du profil utilisateur obtenue par l'approche $\mathcal{E}(\mathcal{R})$ dépend uniquement des relations de la portée du profil utilisateur qui apparaissent également dans la requête initiale. Lorsque le nombre de relations dans la requête initiale augmente, la probabilité qu'une de ces relations apparaisse dans la portée du profil utilisateur augmente également. Les résultats de nos tests (Figure 6.14) montrent qu'en général la couverture du profil de $\mathcal{E}(\mathcal{R})$ augmente avec le nombre de relations dans la requête initiale, mais que ce n'est pas toujours le cas.



a) seuil de pénalité = 0.3



b) seuil de pénalité = 0.7

Figure 6.14 : Couverture du profil utilisateur en fonction du nombre de relations dans la requête initiale

La couverture du profil utilisateur obtenue par l'approche $\mathcal{R}(\mathcal{E})$ ne change pratiquement pas en fonction du nombre de relations dans la requête initiale. Ceci s'explique par le fait que cette approche utilise toujours les Top K prédicats du profil pour enrichir la requête. Si un de ces prédicats est lié à une relation virtuelle qui n'apparaît pas dans la requête, la requête est étendue avec cette relation.

Aucune tendance ne se dégage dans le changement des valeurs de la couverture du profil utilisateur de l'approche $\mathcal{R}/\mathcal{P}$ en fonction du nombre de relations dans la requête initiale. Pour cette approche tout dépend des relations virtuelles dans la requête et de la valeur du seuil de la portée pertinente. En général, pour des petites valeurs de ce seuil (Figure 6.14 a), la couverture du profil utilisateur augmente avec le nombre de relations virtuelles dans la requête initiale. Dans ce cas, lorsque la taille de la portée de la requête initiale augmente, il y a un plus grand nombre de plus courts chemins de jointure à explorer dans la phase de l'expansion de cette requête. En plus, pour des petites valeurs du seuil de la portée pertinente, il y a plus de relations virtuelles de la portée du profil utilisateur qui ne font pas partie de sa portée pertinente. Par conséquent, la probabilité d'étendre la requête initiale avec une de ces relations augmente. Pour des grandes valeurs du seuil de la portée pertinente, la valeur de la couverture du profil utilisateur reste constante (Figure 6.14 b). En effet, dans ce cas il y a moins de relations de la portée du profil utilisateur qui ne font pas partie de la portée pertinente.

Couverture du profil utilisateur en fonction du seuil de pénalité

La seule approche qui utilise le seuil de pénalité est $\mathcal{R}/\mathcal{P}$. Pour les tests que nous avons réalisés, la couverture du profil ne change pas en fonction de ce paramètre parce que tous les prédicats du profil qui peuvent servir à enrichir les réécritures candidates le font avec une pénalité égale à 0.

5.2 Comparaison des temps de réponse

Cette section présente les résultats que nous avons obtenus pour le temps de réponse en fonction des différents paramètres des tests. Nous avons déjà abordé partiellement ces test dans la section 3 pour fixer la taille maximum des requêtes que l'on peut traiter. Nous nous intéressons ici à une analyse plus fine des temps de réponse dans la limite du nombre de relations que l'on peut traiter.

Temps de réponse en fonction du seuil de la portée pertinente

Le temps de réponse de l'approche $\mathcal{E}(\mathcal{R})$ ne change pas en fonction du seuil de la portée pertinente (Figure 6.15). Quelque soit la valeur du seuil de la portée pertinente, cette approche réécrit la requête initiale et ensuite enrichit les réécritures candidates. Son temps de réponse est d'ailleurs presque le même que celui de MiniCon parce que le temps de l'enrichissement des réécritures candidates est négligeable par rapport à celui de la réécriture de la requête. Ceci est une conséquence de l'hypothèse que nous avons faite sur la manière d'enrichir les requêtes qui se fait en y ajoutant une conjonction de prédicats du profil.

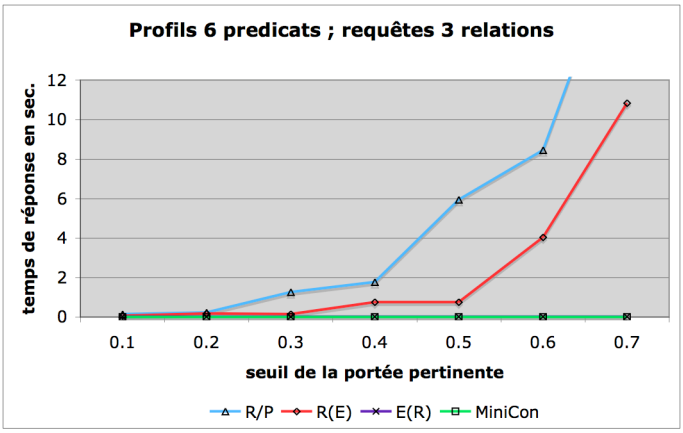


Figure 6.15 : Temps de réponse en fonction du seuil de la portée pertinente

Le temps de réponse des approches $\mathcal{R}/\mathcal{P}$ et $\mathcal{R}(\mathcal{E})$ augmente lorsque la valeur du seuil de la portée pertinente augmente. Pour satisfaire ce seuil, les deux approches étendent la requête initiale avec d'autres relations virtuelles ce qui fait augmenter le temps de réponse de la réécriture de la requête. Le temps de réponse de ces deux approches est nettement supérieur à celui de MiniCon ou de l'approche $\mathcal{E}(\mathcal{R})$.

$\mathcal{R}/\mathcal{P}$ est souvent l'approche la plus coûteuse. En effet, pour le schéma virtuel sur lequel les tests sont réalisés, ajouter une nouvelle relation virtuelle revient souvent à ajouter un seul prédicat de jointure. Autrement dit, très peu de relations virtuelles ont une distance supérieure à 1 par rapport aux requêtes initiales. Par conséquent, la requête enrichie dans $\mathcal{R}(\mathcal{E})$ et la requête étendue dans $\mathcal{R}/\mathcal{P}$ contiennent sensiblement le même nombre de relations virtuelles. L'excédent de temps de réponse de $\mathcal{R}/\mathcal{P}$ par rapport à celui de $\mathcal{R}(\mathcal{E})$ est dû au traitement supplémentaire dans les phases d'expansion et de réécriture de la requête.

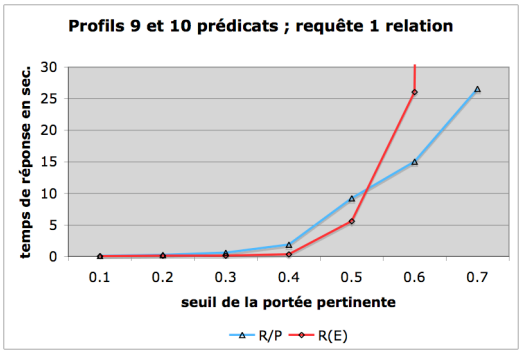


Figure 6.16 : Temps de réponse pour la requête avec 1 relation et les profils de 9 ou 10 prédicats

Dans certains cas le temps de réponse de $\mathcal{R}(\mathcal{E})$ peut être supérieurs à celui de $\mathcal{R}/\mathcal{P}$; c'est le cas des profils ayant 9 ou 10 prédicats. (Figure 6.16). Ce résultat vient du fait que les profils contiennent des prédicats liés à des relations dont la distance par rapport à la requête

initiale est supérieure à 1. Dans ce cas, $\mathcal{R}/\mathcal{P}$ minimise le nombre de relations virtuelles qui sont ajoutées à la requête initiale contrairement à $\mathcal{R}(\mathcal{E})$.

Temps de réponse en fonction du nombre de relations dans la requête initiale

Le temps de réponse de l'ensemble des approches a tendance à augmenter avec le nombre de relations dans la requête initiale (Figure 6.17). Ceci est toujours vrai pour MiniCon et $\mathcal{E}(\mathcal{R})$ qui font la réécriture sur la requête initiale. Étant donné que la réécriture est l'étape la plus coûteuse et qu'elle dépend essentiellement du nombre de relations dans la requête, le temps de réponse de ces deux approches dépend très fortement de ce paramètre.

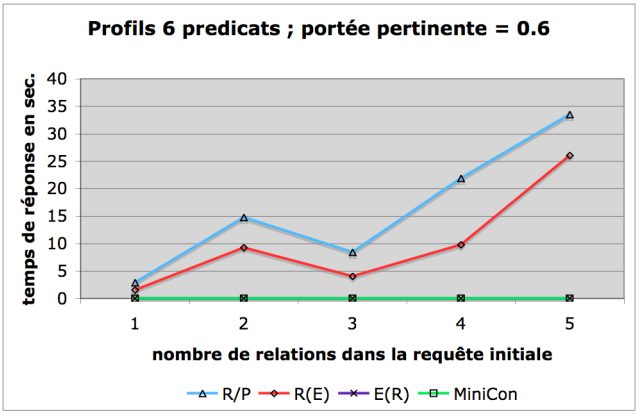


Figure 6.17 : Temps de réponse en fonction du nombre de relations dans la requête initiale

En ce qui concerne $\mathcal{R}/\mathcal{P}$ et $\mathcal{R}(\mathcal{E})$, cette analyse peut leur être associée uniquement si aucun prédicat du profil utilisateur n'est lié aux relations de la requête initiale. Dans le cas contraire, une requête qui initialement contient plus de relations qu'une autre peut en contenir moins que celle-ci après expansion (enrichissement pour $\mathcal{R}(\mathcal{E})$) si une partie du profil utilisateur peut être exprimée sur les relations initiales. Un tel exemple peut être observé sur la Figure 6.17 pour le temps de réponse des requêtes ayant 2 et 3 relations.

Temps de réponse en fonction de la taille du profil utilisateur

Le troisième paramètre dont dépend le temps de réponse est le nombre de prédicats dans le profil utilisateur.

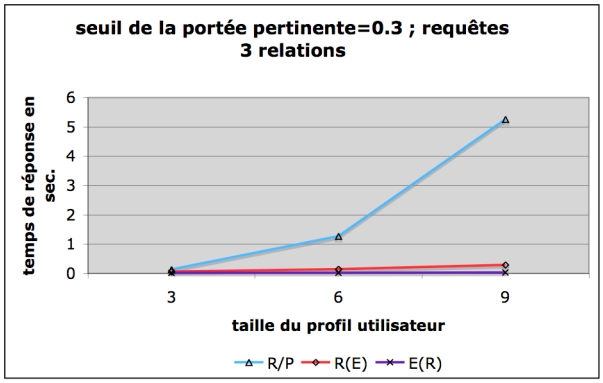


Figure 6.18 : Temps de réponse en fonction de la taille des profils

Le temps de réponse de l'approche $\mathcal{E}(\mathcal{R})$ change peu en fonction de la taille du profil (Figure 6.18). Cette approche ne fait pas d'expansion de la requête initiale et par conséquent la seule étape qui est affectée par la taille du profil utilisateur est l'enrichissement des réécritures candidates. L'hypothèse simplificatrice que nous avons fait sur l'étape

d'enrichissement rend le temps de réponse de $\mathcal{E}(\mathcal{R})$ relativement insensible à la taille du profil utilisateur.

Le temps de réponse des approches $\mathcal{R}/\mathcal{P}$ et $\mathcal{R}(\mathcal{E})$ augmente avec la taille du profil utilisateur (Figure 6.18). Ces deux approches étendent la requête initiale afin de satisfaire le seuil de la portée pertinente. Pour la même valeur de ce seuil, il faut pouvoir tenir compte d'un plus grand nombre de prédicats dans un profil de grande cardinalité que dans un profil de petite cardinalité. Souvent le nombre de prédicats est proportionnel au nombre de relations virtuelles qui doivent être ajoutées à la requête pour pouvoir exploiter ces prédicats ce qui implique une réécriture plus coûteuse. Il est important de noter que nous utilisons une technique très simplifiée pour l'enrichissement des requêtes avec les prédicats des profils utilisateurs. Pour des processus d'enrichissement plus complexes, on peut s'attendre à une augmentation plus nette du temps de réponse de toutes les approches lorsque la taille des profils utilisateurs augmente.

Temps de réponse en fonction du seuil de pénalité

Finalement, le dernier paramètre dont dépend le temps de réponse est la valeur du seuil de pénalité. Ce paramètre n'est utilisé que dans $\mathcal{R}/\mathcal{P}$. Comme le montre la Figure 6.19, le temps de réponse de $\mathcal{R}/\mathcal{P}$ augmente avec la valeur du seuil de pénalité. Ce phénomène s'explique par le fait qu'en choisissant une plus grande valeur du seuil de pénalité, moins de combinaisons de MCDs sont éliminées à chaque niveau de la combinaison des MCDs. Par conséquent, il y a plus d'éléments à tester pendant cette phase.

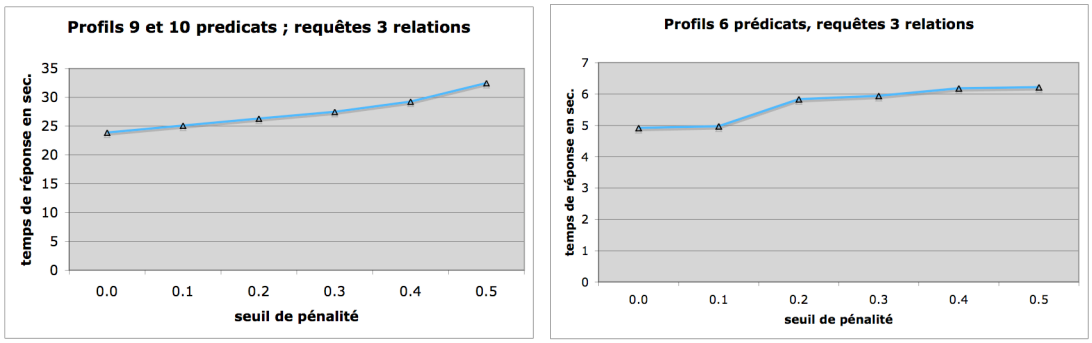


Figure 6.19 : Temps de réponse de $\mathcal{R}/\mathcal{P}$ en fonction du seuil de pénalité

6. Analyse de l'exécution des requêtes

Le second niveau d'évaluation des approches de reformulation de requêtes est le niveau exécution. Les tests de ce niveau ont pour objectif de mesurer la qualité des résultats obtenus par l'exécution des requêtes. Ceci est fait à l'aide de deux métriques qui sont le Rappel et la Précision.

À la différence des résultats de l'évaluation sémantique qui sont présentés en fonction des différents paramètres du benchmark (seuil de la portée pertinente, seuil de pénalité, taille du profil utilisateur et nombre de relations virtuelles dans la requête initiale), nous n'avons pas observé une dépendance entre ces paramètres et les résultats d'exécution. Par conséquent, les résultats dans cette section seront présentés sous la forme de statistiques. Pour obtenir ces statistiques on utilise uniquement les configurations pour lesquelles toutes les approches de reformulation ont retourné des résultats.

6.1 Évaluation du Rappel

La première métrique d'évaluation des résultats de l'exécution des requêtes est le Rappel. Pour une requête donnée Q_u, un profil utilisateur P_u et une approche de reformulation F, le Rappel est égal au pourcentage de résultats pertinents retournés par l'exécution des reformulations de Q_u obtenues par F par rapport au nombre total de résultats pertinents que Q_u aurait du retourner selon P_u. Autrement dit, le Rappel mesure la capacité d'une approche à produire tous les résultats du référentiel de résultats pertinents.

Pour pouvoir comparer les Rappels des différentes approches nous avons besoin d'un référentiel contenant l'ensemble de résultats pertinents. Pour les tests décrits dans ce chapitre, ce référentiel est représenté par les résultats ayant une note supérieure ou égale à 4, obtenus par l'exécution des réécritures de la requête initiale, sans personnalisation. Rappelons que par la construction du benchmark, le Rappel de l'exécution des réécritures de la requête initiale (réécritures obtenues par MiniCon) est de 100% i.e. la requête initiale permet d'obtenir tous les résultats pertinents. À partir de ce référentiel, la perte de Rappel de chaque approche est mesurée. En effet, la personnalisation dans notre contexte se traduit par l'introduction de nouveaux prédicats (restrictions) dans la requête initiale. Ceci implique que le Rappel qui en résulte est inférieur à celui de MiniCon.

Le Tableau 6.16 présente les statistiques de l'évaluation du Rappel. Une case de ce tableau contient le pourcentage des cas dans lesquels le Rappel d'une approche de reformulation est dans un intervalle donné.

Intervalle de Rappel en %	$\mathcal{R}/\mathcal{P}$	$\mathcal{R}(\mathcal{E})$	$\mathcal{E}(\mathcal{R})$
[50, 60)	2,3%		
[60, 70)	10,4%		
[70, 80)	11,6%	0,4%	
[80, 90)	56,8%	0,4%	
[90, 100)	8,5%	52,5%	5,0%
100	10,4%	46,7%	95,0%

Tableau 6.16 : Statistiques du Rappel

Ces statistiques montrent que $\mathcal{E}(\mathcal{R})$ possède la plus petite perte de Rappel. En effet, cette approche possède la plus petite couverture du profil utilisateur et de ce fait ajoute le moins de restrictions supplémentaires à la requête initiale. Il est donc normal d'avoir un Rappel proche de celui de MiniCon.

Comparé à $\mathcal{R}/\mathcal{P}$, $\mathcal{R}(\mathcal{E})$ possède un meilleur Rappel. Étant donné que l'objectif de $\mathcal{R}(\mathcal{E})$ est d'enrichir la requête avec les prédicats de plus grand poids du profil utilisateur et que les poids des prédicats sont les fréquences de leur apparition dans les résultats pertinents, il est naturel que cette approche obtienne un meilleur Rappel. Comme $\mathcal{R}/\mathcal{P}$ privilégie la couverture du profil utilisateur i.e. prend en compte plus de prédicats du profil dans le processus de reformulation de la requête, il ajoute plus de restrictions à la requête initiale et de ce fait perd en Rappel. De plus, cette approche peut utiliser des prédicats de plus petit poids comparés à ceux utilisés par $\mathcal{R}(\mathcal{E})$. Cet effet est provoqué par la différence de la granularité des éléments du profil utilisateur que les deux approches manipulent pendant l'expansion de la requête. $\mathcal{R}/\mathcal{P}$ manipule des relations auxquelles sont liés les prédicats du profil, alors que $\mathcal{R}(\mathcal{E})$ manipule directement les prédicats.

Il est important à noter que la perte de Rappel de $\mathcal{R}/\mathcal{P}$ ne dépasse pas 50% et dans la majorité des cas est inférieure à 20%.

6.2 Évaluation de la Précision

La seconde métrique d'évaluation de l'exécution des requêtes est la Précision. Elle est définie comme le pourcentage de résultats pertinents obtenus par l'exécution des reformulations d'une requête et le nombre total de ces résultats.

Contrairement au Rappel, la Précision des résultats obtenus par l'exécution des réécritures de la requête initiale n'est pas de 100%. Dans ce cas, la Précision obtenue par MiniCon est une valeur de référence que les autres approches de reformulation doivent améliorer étant donné que MiniCon est une approche sans personnalisation.

Le Tableau 6.19 présente les statistiques de la Précision de l'ensemble des approches de reformulation de requêtes.

Intervalle de Précision en %	$\mathcal{R}/\mathcal{P}$	$\mathcal{R}(\mathcal{E})$	$\mathcal{E}(\mathcal{R})$	MiniCon
[40, 50)	21,2%	25,9%	25,9%	25,9%
[50, 60)	10,0%	6,9%	6,9%	6,9%
[60, 70)	25,9%	27,8%	27,8%	27,8%
[70, 80)	37,5%	34,0%	34,0%	34,0%
[80, 90)	5,4%	5,4%	5,4%	5,4%

Tableau 6.17 : Statistiques de la précision

On peut constater qu'à une échelle utilisant des intervalles de 10% de Précision, seule l'approche $\mathcal{R}/\mathcal{P}$ possède des statistiques différentes comparées à celles de MiniCon. La principale cause de ce phénomène est la petite différence entre les valeurs de la Précision des approches pour une même configuration. Ces différences sont suffisamment petites pour être invisibles sur ces statistiques. Pour pallier ce problème, on utilise le gain de Précision d'une approche de reformulation par rapport à MiniCon. Ce gain est égal à la différence entre la Précision de cette approche et celle de MiniCon. Les statistiques du gain de Précision sont représentées dans le Tableau 6.18. Une case de ce tableau contient le pourcentage des cas où le gain de Précision d'une approche est dans un intervalle donné. Certains intervalles ont des valeurs négatives. Ceci correspond aux exécutions pour lesquelles les approches de reformulation de requêtes avec personnalisation ont une moins bonne Précision que MiniCon.

	$\mathcal{R}/\mathcal{P}$	$\mathcal{R}(\mathcal{E})$	$\mathcal{E}(\mathcal{R})$
[8, 10)	2,3%		
[6, 8)	7,3%		
[4, 6)	11,2%		
[2, 4)	25,1%		
[0, 2)	22,8%	29,0%	6,9%
0	0,8%	34,0%	88,0%
(0,-2]	20,8%	37,1%	5,0%
(-2, -4]	7,3%		
(-4, -6]	2,3%		

Tableau 6.18 : Statistiques du gain de précision par rapport à MiniCon

L'approche $\mathcal{E}(\mathcal{R})$ possède une Précision qui dans la majorité des cas (88%) est identique à celle de MiniCon. Cette approche fait très peu de personnalisation i.e. ajoute très peu de nouveaux prédicats du profil à la requête. De ce fait, elle permet d'obtenir sensiblement les mêmes résultats que MiniCon ce qui explique les ressemblances de leur Précision.

Concernant $\mathcal{R}(\mathcal{E})$, dans 29% des cas, elle améliore la Précision des résultats, mais dans 37% des évaluations, sa Précision est moins bonne que celle de MiniCon. Ce phénomène vient des prédicats du profil utilisateur qui sont calculés uniquement par rapport à leur présence dans les résultats pertinents. Il est donc possible que ces prédicats soient satisfaits

également par les résultats non pertinents. Dans le pire des cas il se peut qu'il y ait plus de résultats pertinents qui ne satisfont pas le prédicat que de résultats non pertinents (si le poids du prédicat est inférieur à 1). Dans ce cas, enrichir la requête avec le prédicat en question, dégrade la précision.

Finalement, dans plus de 68% des cas $\mathcal{R}/\mathcal{P}$ améliore la Précision des résultats par rapport à MiniCon et dans un peu plus de 30% elle la dégrade. Cette approche utilise plus de prédicats du profil utilisateur que toutes les autres approches ce qui explique ce gain de Précision.

La dernière mesure sur la Précision est faite pour trouver le pourcentage des cas où chaque approche permet d'obtenir la meilleure valeur de la Précision. Les résultats de cette évaluation sont présentés dans le Tableau 6.19. Ces statistiques montrent que dans près de 70% des cas l'approche $\mathcal{R}/\mathcal{P}$ obtient la meilleure Précision. Pour les autres approches c'est le cas dans environ 20% des exécutions.

$\mathcal{R}/\mathcal{P}$	$\mathcal{R}(\mathcal{E})$	$\mathcal{E}(\mathcal{R})$	MiniCon
69,5%	21,2%	19,3%	19,7%

Tableau 6.19 : Statistiques sur le nombre de fois qu'une approche a obtenu la meilleure Précision

7. Conclusion

Nous avons montré dans ce chapitre l'évaluation des différentes approches de reformulation de requêtes. Cette évaluation a été faite au niveau de la compilation et de l'exécution des requêtes. Les résultats obtenus montrent que les approches qui étendent la requête utilisateur avant de la réécrire ($\mathcal{R}/\mathcal{P}$ et $\mathcal{R}(\mathcal{E})$) sont plus coûteuses en temps de réponse que celles qui réécrivent la requête initiale ($\mathcal{E}(\mathcal{R})$ et MiniCon). En compensation de ce temps de réponse supplémentaire, ces approches ont une meilleure couverture du profil utilisateur et permettent d'obtenir la meilleure Précision dans un plus grand nombre d'exécutions.

Les évaluations que nous avons réalisées ouvrent plusieurs perspectives. Une de ces perspectives est la dérivation d'autres benchmarks à partir de la plateforme de tests afin de faire varier les paramètres que nous avons considérés comme fixés (comme par exemple le nombre de sources disponibles). Une seconde perspective est l'amélioration du benchmark pour que les prédicats du profil utilisateurs tiennent compte non seulement des résultats pertinents, mais également des résultats non pertinents. Cette amélioration permet d'obtenir un profil utilisateur où les prédicats sont plus filtrants pour les résultats non pertinents.

Chapitre 7. Conclusion

Ce chapitre présente la conclusion de la thèse. Les principales contributions de la thèse sont résumées avant de conclure sur les perspectives de recherche à court et à long terme.

1. Résumé des contributions

Les deux problèmes principaux auxquels cette thèse a apporté des contributions sont : (i) la modélisation et la gestion des connaissances qui décrivent l'utilisateur et son contexte et (ii) l'exploitation du profil dans la reformulation de requêtes.

1.1 Contributions de modélisation et gestion de profils et de contextes

Nous avons défini un cadre de référence pour la définition et la gestion de profils et de contextes. Ce cadre de référence est présenté sous la forme d'un méta modèle qui permet de classifier l'ensemble des connaissances décrivant l'utilisateur. Concrètement, nous avons clarifié les concepts de profil, de contexte et de préférences en les étudiant séparément. Un méta modèle a été proposé pour chaque concept illustrant ainsi la typologie des connaissances le constituant. Les relations sémantiques entre les trois méta modèles ont été présentées ce qui a permis d'obtenir une description complète de l'utilisateur.

Tous les méta modèles sont extensibles et peuvent être adaptés à plusieurs applications. Dans leur état actuel, ils permettent d'acquérir l'ensemble des connaissances manipulés par les systèmes que nous avons étudiés. Étant donné que la personnalisation est un domaine en pleine expansion, notre proposition est ouverte et peut facilement prendre en compte les éventuelles nouvelles exigences des systèmes.

Nous avons également proposé deux opérations de gestion des profils et des contextes qui sont l'instanciation et l'appariement. L'instanciation est faite à deux niveaux : (i) instanciation du méta modèle ce qui permet d'obtenir un modèle type et (ii) instanciation d'un modèle type ce qui permet d'obtenir une instance particulière. L'appariement de profils est composé de trois primitives qui permettent de vérifier l'équivalence de deux profils, de mettre en facteur leurs parties communes ou d'isoler leurs différences. Ces primitives s'appliquent aussi bien sur la structure que sur les valeurs des profils.

1.2 Contributions sur la reformulation de requêtes

Nous avons proposé et évalué trois approches de reformulation personnalisé de requêtes dans un contexte distribué. Les deux premières approches sont des approches séquentielles obtenues par composition d'algorithmes de réécriture et d'enrichissement existants. Concrètement, nous avons utilisé l'algorithme de réécriture MiniCon [HP01] et l'approche d'enrichissement de requêtes de Koutrika et Ioannidis [KI04a]. Selon l'ordre dans lequel les deux algorithmes sont appliqués, nous avons proposé une approche réécriture-enrichissement qui réécrit la requête initiale et ensuite enrichit chaque réécriture candidate et une approche enrichissement-réécriture qui consiste à enrichir la requête initiale avant de réécrire la requête enrichie. Les deux approches ont été comparées sur la base de deux métriques que nous avons définies. Ces métriques permettent de mesurer la couverture du profil utilisateur par les prédicats du profil que les approches sont capables de prendre en compte pour enrichir la

requête et l'utilité réelle des prédicats ajoutés à la requête. La comparaison des deux approches a permis de conclure que l'approche enrichissement-réécriture permet d'obtenir une bonne couverture du profil utilisateur, mais peut enrichir la requête avec des prédicats inutiles, alors que l'approche réécriture-enrichissement possède une très bonne utilité, mais souffre en couverture du profil.

Pour gommer les imperfections des deux approches séquentielles, nous avons proposé une approche avancée de reformulation de requêtes. Cette approche comprend quatre phases :(i) expansion de la requête, (ii) recherche des sources contributives, (iii) combinaison des sources contributives et (iv) enrichissement final. Nos contributions ont été faites dans la première et la troisième phase de cette approche ainsi que sur l'évaluation des approches de reformulation de requêtes.

Notre première contribution est un algorithme d'expansion de la requête qui permet de rendre exploitables les prédicats du profil utilisateur. Il fonctionne en deux étapes : choix d'un ensemble de relations virtuelles à ajouter dans la requête et intégrations des relations virtuelles dans la requête. La première phase consiste à trouver les relations virtuelles qui maximisent l'exploitabilité du profil utilisateur tout en minimisant leur nombre. La seconde phase de l'algorithme consiste à intégrer les relations choisies à la requête. Lors de cette phase l'algorithme permet de : préserve la sémantique de la requête initiale, minimiser le nombre de nouvelles relations ajoutées et maximiser la part du profil utilisateur qui peut être exploité sur la requête étendue.

La seconde contribution est une technique de combinaison de sources contributives qui permet de produire uniquement des réécritures candidates pertinentes. Nous avons décrit un algorithme capable de filtrer le plus tôt possible les combinaisons non pertinentes de sources contributives et de produire des solutions qui satisfont un seuil de pertinence.

Finalement, nous avons évalué les trois approches de reformation de requêtes. Les évaluations ont été faites sur un benchmark que nous avons dérivé à partir d'une plateforme de donnée basée sur deux sources de films. Les évaluations ont montré que la personnalisation permet d'obtenir un gain en exploitabilité du profil utilisateur et en précision des résultats obtenus et qu'en contrepartie, il y a un coût à payer pour ce gain qui se traduit par une augmentation du temps de réponse des algorithmes et une perte de résultats pertinents.

2. Perspectives

Les évaluations que nous avons réalisé ont été faites sur un benchmark particulier défini sur une plateforme de données très riche. Une première perspective serait d'extraire d'autres benchmarks ayant des caractéristiques différentes que celui utilisé dans la thèse afin de voir comment les approches de reformulation de requêtes vont réagir à ces modifications. Parmi les paramètres qui peuvent être modifiés, nous pouvons citer les suivants : la taille et la structure du schéma virtuel, le nombre de sources, les définitions des sources, les échantillons de profils et de requêtes, le choix des prédicats des profils utilisateurs, les valeurs des paramètres des algorithmes de reformulation, etc.

Outre le changement des paramètres du benchmark, il serait intéressant de considérer le problème de passage à l'échelle de l'approche, par exemple en fonction la taille des profils utilisateurs. Rappelons, que la limite de l'implémentation actuelle de l'algorithme de réécriture est atteinte lorsque le nombre de relations virtuelles dans la requête dépasse 9 relations. Un travail d'optimisation peut être effectué sur cette implémentation afin de pousser ses limites.

Une seconde perspective consiste à ajouter de nouvelles fonctionnalités à l'approche de reformulation avancée. Nous envisageons deux axes de contribution : la prise en compte de la qualité des données dans le choix des sources contributives et la recherche de techniques permettant de simplifier l'expansion de la requête. Actuellement, seul le centre d'intérêt de l'utilisateur est pris en compte dans la reformulation des requêtes. Or une des phases de cette reformulation est le choix des sources contributives et la dimension Qualité du profil utilisateur contient des préférences sur la qualité des sources. La question qui se pose naturellement dans ce contexte est comment intégrer la qualité dans le choix des sources ? Une autre contribution possible est d'éviter de lancer le calcul des plus courts chemins entre les relations de la requête initiale et celles de la portée pertinente autant de fois qu'il y a de relations dans la requête. Nous avons l'intuition qu'en regroupant les relations de la requête en un seul nœud dans le graphe du schéma virtuel, une seule recherche de plus courts chemins suffirait. Cette idée nécessiterait de restructurer le graphe du schéma virtuel à chaque fois qu'une relation est ajoutée dans la requête. Il faut donc analyser le rapport entre le gain obtenu en calcul des plus courts chemins et la perte en restructuration du graphe.

La troisième perspective immédiate est liée à l'exploitation des reformulations produites par les approches de reformulation de requêtes. Toutes les approches proposées dans cette thèse produisent un ensemble de reformulations candidates pour chaque requête. Souvent ces reformulations ne permettent pas d'obtenir les mêmes résultats ; elles interrogent différentes sources, sont enrichies avec différents prédicats du profil utilisateur etc. Notre objectif est double : d'une part, nous voulons trouver des méthodes efficaces pour comparer les reformulations entre elles afin de les ordonner et, d'autre part, nous voulons élaborer des stratégies d'exécution des reformulations (ex. toutes en parallèle, séquentiellement en s'arrêtant à la première qui donne des résultats, etc.).

Les perspectives que nous venons de présenter sont des perspectives immédiates liées au travail décrit dans cette thèse. Ce travail ouvre également des perspectives à plus long terme que nous présentons maintenant.

Le modèle de profil que nous avons présenté dans cette thèse propose une riche palette de caractéristiques de l'utilisateur. Même si prendre en compte tout le contenu du profil utilisateur paraît difficile, à long terme il serait intéressant d'étudier des techniques pouvant prendre en compte une grande partie du profil. La recherche de telles techniques est une de nos perspectives.

Les approches que nous avons présentées traitent la reformulation de requêtes qui est une des étapes du cycle de vie d'une requête. La personnalisation peut intervenir à chacune de ces étapes avec des objectifs différents. Par exemple, l'optimisation de la requête peut être faite dans le but de satisfaire les préférences de l'utilisateur en temps de réponse, alors que l'affichage des résultats peut permettre de tenir compte des préférences de l'utilisateur sur le nombre maximum de résultats qu'il souhaite obtenir. La perspective de personnaliser chacune des étapes du cycle de vie de la requête est alors fortement liée à celle de la prise en compte des autres dimensions du profil utilisateur.

Références

[APV02] Dell'Acqua P., Moniz Pereira L., Vitoria A., *User Preference Information in Query Answering*, In Proceedings of the 5th International Conference on Flexible Query Answering Systems, Copenhagen, Denmark, p. 163-173, 2002.

[ARM+06] Al-Hezmi A., Rebahi Y., Magedanz T., Arbanowski S., *Towards an Interactive IPTV for Mobile Subscribers*, In Proceedings of the International Conference on Digital Telecommunications, ICDT apos'06', Côte d'Azur, France, p. 45-45, 2006.

[AS94] Agrawal R., and Srikant R., *Fast algorithms for mining association rules*, In Proceedings of the International Conference on Very Large DataBases (VLDB), Santiago, Chile, p. 487-499, 1994.

[AS99] Amato G., Straccia U., *User Profile Modeling and Applications to Digital Libraries*, In Proceedings of the Third European Conference on Research and Advanced Technology for Digital Libraries, Paris, France, p. 184-197, 1999.

[AY02] Aggarwal C., Yu P., *An automated system for web portal personalization,* In Proceedings Of 28Th International Conference On Very Large Data Bases (Vldb'02) Hong Kong China, p. 1031-1040, 2002.

[BCD+07] Bouzeghoub M., Calabretto S., Denos N., Harrathi R., Kostadinov D., Nguyen A-T, Peralta V., *Accès personnalisé aux informations : approche dirigée par la qualité*, Dans les actes des 25e Congrès INFORSID, Perros-Guirec, France, p. 105-120, 2007.

[Ben02] Bengoetxea E., *Inexact Graph Matching Using Estimation of Distribution Algorithms*. Ph.D. thesis, ENST. 2002.

[BHL01] Berners-Lee T., Hendler J., Lassila O., *The semantic web*, Scientific American, 2001.

[BHK98] Breese J. S., Heckerman D., Kadie C., *Empirical analysis of predictive algorithms for collaborative filtering*, Technical report MSR-TR-98-12, Microsoft research, Redmond, WA 98052, 1998.

[BK05] Bouzeghoub M., Kostadinov D., *Personnalisation de l'information: aperçu de l'état de l'art et définition d'un modèle flexible de profils*, Dans les actes de la seconde édition de la Conférence en Recherche d'Infomations et Applications (CORIA), Grenoble, France, p. 201-218, 2005.

[BKS01] Borzsonyi S., Kossmann D., Stocker K., *The Skyline Operator*, In Proceedings of the IEEE Conference on Data Engineering (ICDE), Heidelberg, Germany, p. 421-430, 2001.

[BM98] Büchner A., Mulvenna M., *Discovering internet marketing intelligence through online analytical web usage mining*, SIGMOD Record 2, p. 54-61, 1998.

[BP95] Bosc P., Pivert O., *SQLf : A Relational Database Language for Fuzzy Querying*, IEEE Transactions on Fuzzy Systems, vol. 3, No 1, p. 1-17, 1995.

[BR02] Bright, L., Raschid L. *Using Latency-Recency Profiles for Data Delivery on the Web*, In Proceedings of the 28th Conference on Very Large Data Bases, Hong Kong, China, p. 550-561, 2002.

[BRS00] Bradley K., Rafter R., Smyth B., *Case-Based User Profiling for Content Personalisation*, In Proceedings of the International Conference on Adaptive Hypermedia and Adaptive Web-based Systems, Trento, Italy, p. 62-72, 2000.

[Bun97] Bunke H., *On a relation between graph edit distance and maximum common subgraph*, Pattern Recognition Letters, p. 689-694, 1997.

[CDE+05] Cranor L., Dobbs B., Egelman S., Hogben G., Humphrey J., Langheinrich M., Marchiori M., Presler-Marshall M., Reagle J., Schunter M., *The Platform for Privacy Preferences 1.1 (P3P1.1) Specification*. W3C Working Draft 1 July (2005), http://www.w3.org/TR/2005/WD-P3P11-20050701/

[CGFZ03] Cherniack M., Galvez Ed., Franklin M., Zdonik St., *Profile-Driven Cache Management*, In Proceedings of the 19th International Conference on Data Engineering (ICDE), Bangalore, India, p. 645-656, 2003.

[Cha00] Chan. P., *Constructing Web User Profiles: A Non-invasive Learning Approach,* In Web Usage Analysis and User Profiling, LNAI 1836, Springer-Verlag, p. 39-55, 2000.

[Cho02] Chomicki J., *Querying with Intrinsic Preferences*, In Proceeding of the 8th International Conference on Extending Database Technology (EDBT), Prague, Czech Republic, p. 34-51, 2002.

[CLL01] Calvanese, D., Lembo D., Lenerini M. *Survey on methods for query rewriting and query answering using views,* Technical report, University of Rome, Roma, Italy, 2001.

[CLM02] Cranor L., Langheinrich M., Marchiori M., *A P3P Preference Exchange Language 1.0.*, W3C Working Draft, 2002. http://www.w3.org/TR/P3P-preferences/

[CN04] Ciro S., Newton V., *Use Reformulated Profile in Information Filtering*, In Proceedings of the AAAI Workshop on Semantic Web Personalization, San Jose, California, 2004.

[CS01] Chen Y., Shahabi C., *Automatically Improving the Accuracy of User Profiles with Genetic Algorithm*, In Proceeding of the IASTED International Conference on Artificial Intelligence and Soft Computing, 2001.

[CT02] Torlone R., Ciaccia P., *Which Are My Prefered Items?*, Workshop on Recommendation and Personalization in eCommerce (RPEC 2002), Malaga, Spain, 2002.

[DDS02] Dorigo M., Di Caro G., Sampls M., *Ant algorithms*, 3rd International Workshop on Ants Algorithms (ANTS), Brussels, Belgium, 2000

[DEF+05] Van Dijk J.A.G.M., Ebbers W.E., Fennis B.M., van der Geest T.M., Loorbach N.R., Pieterson W.J., Steehouder M.F., Taal E., de Vries P.W., *Alter Ego: State of the art on user profiling*. technical report, University of Twente, 2005.

[Dem02] K., L., Dempski, *Real Time Television Content Platform: Personalized Programming Over Existing Broadcast Infrastructures.* Accenture Technology Labs, Chicago, The Second Workshop on Personalization in Future TV (TV '02), Second International Conference on Adaptive Hypermedia and Adaptive Web-based System USA, 2002.

[DG97] Duschka O., and Genesereth M., *Answering Recursive Queries Using Views*, In Proceedings of the 16th ACM SIGACT-SIGMOD-SIGART Conference on Principles of Database Systems, PODS, Tucson, AZ, p. 109-116, 1997.

[DJBW03] Dumais S., Joachims T., Bharat K., Weigend A., *Sigir 2003 workshop report: implicit measures of user interests and preferences*, SIGIR Forum, p. 50-54, 2003.

[DLR77] Dempster A., Laird N., Rubin D., *Maximum likelihood from incomplete data via the EM*

algorithm, Journal of the Royal Statistical Society (1977) 39: p. 1-38,1977.

[DM05] Dai H., Mobasher B., *Integrating semantic knowledge with web usage mining for personalization,* In Web Mining: Applications And Techniques. Anthony Scime (Ed.), 2005.

[EP00] Eisenstein, J., Puerta A., *Adaptation in Automated User-Interface Design*, In Proceedings of the International Conference on Intelligent User Interfaces, LA, USA, p. 74-81, 2000.

[ES95] Sorensen H., Mc Elligott M., *PSUN : A Profiling System for Usenet News*, In Proceedings of the Intelligent Information Agents Workshop, Conference on Information and Knowledge Management (CIKM'95), Baltimore, 1995.

[EV03] Eirinaki E., Vazirgiannis M., *Web mining for web personalization,* ACM Transaction on Internet Technology, p. 1-27, 2003.

[EVV03] Eirinaki M., Vazirgiannis M., Varlamis I ., *Sewep: using site semantics and a taxonomy to enhance the web personalization process,* In Proceedings Of The Ninth Acm Sigkdd International Conference On Knowledge Discovery And Data Mining Washington Dc USA, p. 99-108, 2003.

[Fer95] Ferguson S., *BEAGLE: A genetic algorithm for Information Filter Profile Creation,* Technical Report CS-692, University of Alabama, 1995.

[FS01] Ferreira J., Silva A., *MySDI: A Generic Architecture to Develop SDI Personalised Services*, In Proceedings of the 3rd International Conference on Enterprise Information Systems, Setubal, Portugal, p. 262-270, 2001.

[FV01] Fernandez M., Valiente G., *A graph distance metric combining maximum common subgraph and minimum common supergraph*. Pattern Recognition Letters, p.753-758, 2001.

[GL94] Gaasterland T., Lobo J., *Qualified Answers that Reflect User Needs and Preferences*, In Proceedings of the 20th Very Large Data Bases (VLDB) Conference, Santiago de Chile, Chile, p. 309-320, 1994.

[GM05] Germanakos P., Mourlas C., *Adaptation and Personalization of Web-based Multimedia Content,* Proceedings of the Workshop on 'Personalization for e-Health' of the 10th International Conference on User Modeling (UM'05), Edinburgh, July 29, 2005, p. 67-70, 2005.

[Grc04] Grcar M., *User profiling: Collaborative filtering ,* SIKDD 2004 at multiconference IS 2004*,* Ljubljana, Slovenia, 2004.

[Hal01] Halevy A.Y., *Answering Queries Using Views: A Survey*, The VLDB Journal, Vol. 10, p. 270-294, 2001.

[HCO+02] Huang Z., Chung W., Ong T., Chen H., *A Graph-based Recommender System for Digital Library,* In Proceedings of the 2nd ACM/IEEE-CS joint conference on Digital libraries, Portland, Oregon, USA, p. 65-73, 2002.

[HK06] Han J., Kamber M., *Data mining: concepts and techniques*, Jim Gray (Ed.). Morgan Kaufmann, 2006.

[Hof99] Hofmann T., *Probabilistic latent semantic indexing,* In Proceedings Of The 22Nd International Acm Sigir Conference On Research And Development In Information Retrieval, Berkeley, Ca, Usa, p. 50-57, 1999.

[HP01] Halevy A., and Pottinger R., *MiniCon: A scalable algorithm for answering queries using views*, Very Large Data Bases Journal, Vol. 10, Num. 2-3, p. 182-198, 2001.

[HRO06] Halevy A., Rajaraman A., Ordille J., *Data integration: the teenage years*, In Proceedings Of The 32Nd International Conference On Very Large Data Bases (Vldb'06), Seoul, Korea, p. 9-16, 2006.

[HRW92] Hwang F. K., Richards D. S., Winter P., *The Steiner Tree Problem*, Elsevier, North-Holland, 1992.

[IMDb07] Intenet Movie Database, Inc.: "The Intenet Movie Database", Web site, URL: http://www.imdb.com/interfaces, last accessed on July 9th, 2007.

[JPR+05] Jaudoin H., Petit J-M., Rey C., Schneider M., Toumani F., *Query rewriting using views in presence of value constraints*, In Proceedings of the 2005 International Workshop on Description Logics (DL2005), Edinburgh, Scotland, UK, p. 112-119, 2005.

[JZM04] Jin X., Zhou Y., Mobasher B., *A unified approach to personalization based on probabilistic latent semantic models of web usage and content*, In Proceedings Of The Aaai 2004 Workshop On Semantic Web Personalization (Swp'04). 2004.

[JZM05] Jin X., Zhou Y., Mobasher B., *A maximum entropy web recommendation system: combining collaborative and content features,* In Proceedings Of The Eleventh Acm Sigkdd International Conference On Knowledge Discovery And Data Mining Chicago Illinois, USA, p. 612-617, 2005.

[KBL07a] Kostadinov D., Bouzeghoub M., Lopes S., *Query Rewriting Based on User's Profile Knowledge*, Dans les actes des 23e Journées Bases de Données Avancées (BDA), Marseille, France, 2007.

[KBL07b] Kostadinov D., Bouzeghoub M., Lopes S., *Accès personnalisé à des sources de données multiples : évaluation de deux approches de reformulation de requêtes*, Dans les actes des 25e Congrès INFORSID, Perros-Guirec, France, p. 73-88, 2007.

[Kie02] Kießling W., *Foundations of Preferences in Database Systems*, In Proceedings of the 28th Conference on Very Large Data Bases, Hong Kong, China, p. 311-322, 2002.

[Kie05] Kießling W., *Preference Queries with SV-Semantics*, In Proc. of the 11th International Conference on Management of Data (COMMAD 2005), Goa, India, p. 15-20, 2005.

[KI04a] Koutrika G., Ioannidis Y. E., *Personalization of Queries in Database Systems*, In Proceedings of the 20th International Conference on Data Engineering, Boston, Massachusetts, USA, p. 597-608, 2004.

[KI04b] Koutrika G., Ioannidis Y. E., *Rule-based query personalization in digital libraries*, Int. Journal on Digital Libraries 4(1): 60-63, p. 60-63, 2004.

[KI05a] Koutrika G., Ioannidis Y. E., *Constrained Optimalities in Query Personalization*, In Proceedings of the ACM SIGMOD, Baltimore, Maryland, USA, p. 73-84, 2005.

[KI05b] Koutrika G., Ioannidis Y. E., *Personalized Queries under a Generalized Preference Model*, In Proceedings of the 21st International Conference on Data Engineering (ICDE 2005), Tokyo, Japan, p. 841-852, 2005.

[KKB+05] Kim C. S., Kim D. Y., Bae T. M., Ro Y. M., *Versatile video on demand system*, In IConsumer Electronics (ISCE 2005), Proceedings of the Ninth International Symposium on Volume, Issue 19, p. 441-444, 2005.

[Kna05] Knappe R., *Measures of semantic similarity and relatedness for use in ontology-Based information retrieval*, Ph.D Thesis, Roskilde University, Department of Communication, Journalism and Computer Science. Chapter 5, page 69, 2005.

[KPS+04] Kostadinov D, Peralta V., Soukane A., Xue X., *Système Adaptatif d'aide à la Génération*

de Requêtes de Médiation, Démonstration aux 20e Journées Bases de Données Avancées (BDA), Montpellier, France, p. 351-355, 2004.

[KPS+05] Kostadinov D, Peralta V., Soukane A., Xue X.: *Intégration de données hétérogènes basée sur la qualité*, Dans les actes des 23e Congrès INFORSID, Grenoble, France, p. 471-486, 2005.

[KT03] Kelly D., Teevan J., *Implicit feedback for inferring user preference: a bibliography*, SIGIR Forum, p. 18-28, 2003.

[LBE+98] Loupy C., Bellot P., El-Bze M., Marteau P.-F., *Query Expansion and Classification of Retrieved Documents*, In Proceedings of TREC-7, Gaithersburg, Maryland, USA, p. 382-389, 1998.

[Len02] Lenzerini M., *Data integration: a theoretical perspective,* In Proceedings Of The 21St Acm Sigmod-Sigact-Sigart Symposium On Principles Of Database Systems (Pods'02), Madison, Wisconsin, p. 233-246, 2002.

[Lie95] Lieberman H., *Letizia: An Agent That Assists Web Browsing,* In Proceedings of the Fourteenth International Joint Conference on Artificial Intelligence, p. 924-929, 1995.

[LL87] Lacroix M., Lavency P., *Preference: Putting More Knowledge into Queries*, In Proceeding of the 13th Conference on Very Large Data Bases, Brighton, England, p. 217-225, 1987.

[LNR+01] Liu C., Wu J., Yu C., Nakajima H, Rishe N. D., Yang Q., Zhang W., *Efficient Processing of Nested Fuzzy SQL Queries in a Fuzzy Database*, IEEE Transactions on Knowledge and Data Engineering, vol. 13, No 6, November/December 2001.

[LRO96] A. Y. Levy, A. Rajaraman and J. J. Ordille, *Querying Heterogeneous Information Sources Using Source Descriptions*, In Proceedings of the 22nd Very Large Data Bases Conference, Bombay, India, p. 251-262, 1996.

[LS01] Shearin S., LiebermanH, *Intelligent Profiling by Example*, In Proceedings of the 2001 International Conference on Intelligent User Interfaces, Santa Fe, USA, p. 145-151, 2001.

[MC07] Michlmayr E., Cayzer S., *Learning User Profiles from Tagging Data and Leveraging them for Personal(ized) Information Access,*WWW2007, Banff, Canada, p. ,2007

[MDS01] Middleton S., De Roure D., Shadbolt N., *Capturing Knowledge of User Preferences: ontologies on recommender systems*, In Proceedings of the First International Conference on Knowledge Capture, Victoria, B.C. Canada, p. 100-107, 2001.

[Mes95] Messmer B., *Graph matching algorithms and application*, Ph.D. thesis, University of Bern, 1995.

[ML06] GroupLens Research: "movielens: helping you to find the right movies". Web site, ULR: http://movielens.umn.edu, last accessed on July 9th, 2006.

[Mob05] Mobasher B., *Web Usage mining and personalization,* In Practical Handbook Of Internet Computing. Singh Munindar P. (Ed.), 2005.

[Mob07] Mobasher B., *Data mining for personalization,* In The Adaptive Web: Methods And Strategies Of Web Personalization. Brusilovsky P. and Kobsa A. and Nejdl W. (Ed.), p. 90-105, 2007.

[MSR04] Middleton, S. E., Shadbolt, N. R., De Roure, D. C., *Ontological User Profiling in Recommender Systems*, ACM Transactions of Information Systems, p. 54-88, 2004.

[MVL99] Martin-Bautista M., and Vila M., Larsen H., *FUZZY Genetic Algorithm Approach to an Adaptive Information Retrieval Agent*, Journal of the American Society for Information Science, p. 760-771, 1999.

[MVL00] Martin-Bautista M., and Vila M., Larsen H., *Building adaptive user profiles by a genetic fuzzy classifier with feature selection,* In Proceedings of the IEEE Conference on Fuzzy Systems vol.1, San Antonio, Texas, p. 308-312, 2000.

[MWT04] Mushtaq N., Werner P., Tolle K., Zicari R., *Building and Evaluating Non-Obvious User Profiles for Visitors of Web Site,* In Proceedings of the IEEE International Conference on E-Commerce Technology (CEC'04), p. 9-15, 2004.

[NDB06] Nguyen A.-T., Denos N., Berrut C., *Exploitation des données « disponibles à froid » pour améliorer le démarrage à froid dans les systèmes de filtrage d'information*, Dans les Actes du XXIVème Congrès INFORSID, Hammamet, Tunisie, p. 81-95, 2006.

[NFS98] Naumann, F., Freytag J.C., Spiliopoulou M., *Quality Driven Source Selection Using Data Envelope Analysis*, In Proceedings of the MIT Conference on Information Quality (IQ'98), Cambridge, USA, p. 137-152, 1998.

[Paz99] Pazzani M. J., *A Framework for Collaborative, Content-Based and Demographic Filtering,* technical report, University of California, Irvine, 1999.

[Per07a] Peralta V., *Extraction and Integration of MovieLens and IMDb Data*, Technical Report, Laboratoire PRiSM, Université de Versailles, Versailles, France, 2007.

[Per07b] Peralta V., *A Benchmark for Data Personnalisation*, Technical Report, Laboratoire PRiSM, Université de Versailles, Versailles, France, 2007.

[PML05] Pampapathi R., Mirkin B., Levene M., A Review of the Technologies and Methods in Profiling and Profile Classification, Review 2005.

[Pot04] Potonniée O., *A decentralized privacy-enabling TV personalization framework*, 2nd European Conference on Interactive Television: Enhancing the Experience, Brighton, U.K., 2004.

[PRGM03] Pigeau A., Raschia G. , Gelgon M., Mouaddib N., Saint-Paul R., *A fuzzy linguistic summarization technique for TV recommender systems,* Fuzzy Systems, FUZZ '03 The 12th IEEE International Conference, p. 743-748, 2003.

[PG99] Pretschner A., Gauch S., *Ontology Based Personalized Search*, In Proceedings of the 11th IEEE International Conference on Tools with Artificial Intelligence (ICTAI'99), Chicago, USA, p. 391-398, 1999.

[Qia96] Qian X., *Query folding*, In Proceedings of the Twelfth In-ternational Conference on Data Engineering (ICDE), New Orleans, Louisiana, p. 48-55, 1996.

[RBS00] Rafter R., Bradley K., Smyth B., *Automated Collaborative Filtering Applications for Online Recruitment Services*, Lecture Notes In Computer Science; Vol. 1892, Proceedings of the International Conference on Adaptive Hypermedia and Adaptive Web-Based Systems, Trento, Italy, p. 363-368, 2000.

[RL06] Rocacher D., Liétard L., *Préférences et quantités dans le cadre de l'interrogation flexible: sur la prise en compte d'expressions quantifiées*, Dans les actes des 22e Journées Bases de Données Avancées (BDA), Lille, France, 2006.

[RM02] Raschia G., Mouaddib N., *A fuzzy set-based approach to database summarization*, Journal of Fuzzy Sets and Systems, vol. 129, n. 2, p. 137–162, July 2002.

[Roc03] Rocacher D., *On fuzzy bags and their application to flexible querying*, Fuzzy sets and

Systems, 140 (1), p. 93-110, 2003.

[SCZ05] Song R.,Chen E., Zhao M., *SVM Based Automatic User Profile Construction for Personalized Search,* In Proceedings of the International Conference on Intelligent Computing ICIC , China, p. 475-484, 2005.

[SF06] Schickel-Zuber V., Faltings B., *Inferring User's Preferences using Ontologies*, In Proceeding of AAAI 2006 (2006), p. 1413-1418, 2006.

[SHY04] Sugiyama K., Hatano K., Yoshikawa M., *Adaptive web search based on user profile constructed without any effort from users*, In Proceeding of the 13th International World Wide Web Conferences (WWW), New York, USA, p. 675-684, 2004.

[SKR01] Schafer J., Konstan J., Riedl J., *E-Commerce Recommendation Applications,* Data Mining and Knowledge Discovery, Volume 5, Numbers 1-2, 4 January 2001 .

[SL01] Shearin S., Lieberman H., *Intelligent Profiling by Example*, In Proceedings of the International Conference on Intelligent User Interfaces, USA, p. 145-151, 2001.

[SMS07] Shoval P., Maidel V., Shapira B., *An Ontology- Content-Based Filtering Method*, I.Tech-2007 - Information Research and Applications, 2007.

[SP04] Schmidt C., Parachar M., *Enabling Flexible Queries with Guarantees in P2P Systems*, IEEE Internet Computing , vol. 8, No 3, May/June 2004.

[SPT06] Siberski W., Pan J., Thaden U., *Querying the Semantic Web with Preferences*, In Proceeding of the 5th International Semantic Web Conference, Athens, GA, USA, p. 612-624, 2006.

[Sus93] Sussna, M., *Word Sense Disambiguation for Free-text Indexing Using a Massive Semantic Network,* In Proceedings of CIKM, p. 67-74, 1993.

[TM80] Takahashi H., Matsuyama A., *An approximate solution for the Steiner problem in graphs*, Mathematica Japonica, 24, p. 573-577, 1980.

[TPC07] Transaction Processing Performance Council. URL: http://www.tpc.org/, last accessed on September 2007.

[TREC07] Text REtrival Conference (TREC).URL: http://trec.nist.gov/, last accessed on September, 2007.

[VFC+06] Vallet D., Fernández M., Castells P., Mylonas P., Avrithis Y., *Personalized Information Retrieval in Context*, In Proceeding of the 3rd International Workshop on Modeling and Retrieval of Context (MRC 2006) at the 21st National Conference on Artificial Intelligence (AAAI 2006). Boston, USA, 2006.

[VRM+06] Vidal, M.E., Raschid L., Marquez N., Cardenas M., Wu Y., *Query Rewriting in the Semantic Web*, In Proceedings of the 22nd International Conference on Data Engineering Workshops, ICDEW 2006, Atlanta, GA, USA, 2006.

[ZTB07] Zemirli N., Tamine-Lechani L., Boughanem M., *Présentation et évaluation d'un modèle d'accès personnalisé à l'information basé sur les diagrammes d'influence*, Dans les Actes du XXVème congrès INFORSID, Perros-Guirec, France, p. 89-104, 2007.

[ZXH98] Zaïane O., Xin M., Han J., *Discovering web access patterns and trends by applying olap and data mining technology on web logs,* In Proceedings of the Advances in Digital Libraries Conf. (ADL'98), Santa Babara, p. 19-29, 1998.

Annexe 1 : Données des premiers tests

Schéma virtuel :

```
VOYAGE(idV, prix, lieu_depart, lieu_arrivee, nbre_jours, date_depart, heure,
    type_sejour, type_formule, idT, idH)

TRANSPORT(idT, moyen, type_trajet, comfort)

HOTEL(idH, nbre_etoiles, nom, region, ville, restaurant)
```

Profils utilisateurs :

Profil P1 :

```
{ TRAVEL.nbDays > 7                  1.0  (c)
travel.departure = 'Toulouse'        0.8  (d)
transport.mean = 'plane'             0.7  (e)
transport.wayType = 'direct'         0.6  (f)
hotel.nbStars > 3                    0.5  (g)
travel.tripType <>'circuit'          0.5  (h)
transport.comfort > 2                0.4  (i)
travel.departure = 'Paris'           0.3  (j)
hotel.region = 'centre city'         0.2  (k) }
```

Profil P2 :

```
{VOYAGE.lieu_depart='Paris'          1.0
 VOYAGE.lieu_arrivee='Madrid'        0.9
 VOYAGE.lieu_arrivee='Barcelone'     0.7
 TRANSPORT.moyen='avion'             0.7
 VOYAGE.type_formule='week end'      0.6
 VOYAGE.date_depart='05/05/2006'     0.6
 TRANSPORT.moyen='train'             0.4
 VOYAGE.heure='10H10'                0.4
 TRANSPORT.type_trajet='direct'      0.2
 VOYAGE.prix<1000                    0.2 }
```

Profil P3 :

```
{VOYAGE.lieu_arrivee='Barcelone'     1.0
 VOYAGE.type_formule='week end'      1.0
 VOYAGE.lieu_depart='Lyon'           0.8
 VOYAGE.prix<800                     0.8
 TRANSPORT.moyen='train'             0.7
 TRANSPORT.moyen='avion'             0.3
 VOYAGE.lieu_depart='Paris'          0.2
 TRANSPORT.comfort>2                 0.1
}
```

Profil P4 :

```
{HOTEL.nbre_etoiles=2                0.9
 VOYAGE.lieu_depart='Paris'          0.9
 HOTEL.region='banlieu'              0.8
 VOYAGE.nbre_jours=2                 0.7
 TRANSPORT.moyen='train'             0.6
 TRANSPORT.moyen='car'               0.5
 TRANSPORT.comfort=3                 0.4
 VOYAGE.type_formule='sejour'        0.3
 VOYAGE.type_formule='thalasso'      0.3
 VOYAGE.prix<800                     0.2
 HOTEL.restaurant='oui'              0.1
}
```

Définitions des sources :

```
S1:HOTELSDUMONDE(idH, nbre_etoiles, nom, region, ville, restaurant) :-
    HOTEL(idH, nbre_etoiles, nom, region, ville, restaurant).

S2:TRANSPORTAERIEN(idT, lieu_depart, lieu_arrivee, date_depart, heure, moyen,
   type_trajet, comfort) :-
   TRANSPORT(idT, 'avion', type_trajet, comfort),
   VOYAGE(idV,prix,lieu_depart, lieu_arrivee, nbre_jours, date_depar, heure,
      type_sejour, type_formule, idT, idH).

S3:SNCF(idT,lieu_depart,lieu_arrivee, date_depart, heure, moyen, type_trajet,
   comfort) :-
   TRANSPORT(idT, 'train', type_trajet, comfort),
   VOYAGE(idV, prix, lieu_depart, lieu_arrivee, nbre_jours, date_depart, heure,
      type_sejour, type_formule, idT, idH).

S4:VOYAGERPARTOUT(idT, lieu_depart, lieu_arrivee, date_depart, heure, moyen,
   type_trajet, comfort) :-
   TRANSPORT(idT, moyen, type_trajet, comfort),
   VOYAGE(idV, prix, lieu_depart, lieu_arrivee, nbre_jours, date_depart, heure,
      type_sejour, type_formule, idT, idH).

S5:PROMOVACANCES(idV, prix, lieu_depart, lieu_arrivee, nbre_jours, date_depart, heure,
   type_sejour, type_formule, moyen, nom, nbre_etoiles, restaurant) :-
   TRANSPORT(idT, moyen, type_trajet, comfort),
   VOYAGE(idV, prix, 'Paris', lieu_arrivee, nbre_jours, date_depart, heure,
      type_sejour, type_formule, idT, idH),
   HOTEL(idH, nbre_etoiles, nom, region, lieu_arrivee, restaurant).

S6:LYONVACANCES(idV, prix, lieu_depart, lieu_arrivee, nbre_jours, date_depart, heure,
   type_sejour, type_formule, moyen, nom,nbre_etoiles,restaurant) :-
```

```
TRANSPORT(idT, moyen, type_trajet, comfort),
VOYAGE(idV, prix,'Lyon', lieu_arrivee, nbre_jours, date_depart, heure,
   type_sejour, type_formule, idT,idH),
HOTEL(idH, nbre_etoiles, nom, region, lieu_arrivee, restaurant).
```

Echantillon de requêtes initiales pour le test :

Q1 :
```
SELECT idV, prix, V.lieu_depart, V.lieu_arrivee, T.moyen, T.comfort
FROM VOYAGE V, TRANSPORT T
WHERE V.idT = T.idT and V.lieu_arrivee='Madrid' and V.nbre_jours=4;
```

Q2 :
```
SELECT idV, V.prix, V.lieu_depart, V.lieu_arrivee, V.date_depart
FROM VOYAGE V;
```

Q3 :
```
SELECT idV, V.prix, V.lieu_depart, V.lieu_arrivee, T.moyen, H.nom, T.comfort,
  H.region
FROM VOYAGE V, TRANSPORT T, HOTEL H
WHERE V.idT = T.idT and V.nom=H.nom and V.lieu_arrivee='Venise' and
  V.nbre_jours<13;
```

Q4 :
```
SELECT idV, V.prix, V.lieu_depart, V.lieu_arrivee, T.moyen, T.type_trajet
FROM VOYAGE V, TRANSPORT T
WHERE V.idT=T.idT and V.type_sejour='pension complete' and T.moyen ='train';
```

Q5 :
```
SELECT idV, prix, V.lieu_depart, V.lieu_arrivee, V.date_depart, V.type_sejour
FROM VOYAGE V
WHERE V.lieu_arrivee='Rome' and V.prix<700;
```

Q6 :
```
SELECT idV, V.prix, V.lieu_depart, V.lieu_arrivee, T.moyen, T.type_trajet
FROM VOYAGE V, TRANSPORT T
WHERE V.idT = T.idT and V.lieu_depart='Paris' and V.lieu_arrivee='Madrid' and
  V.type_sejour='hotel et trajet';
```

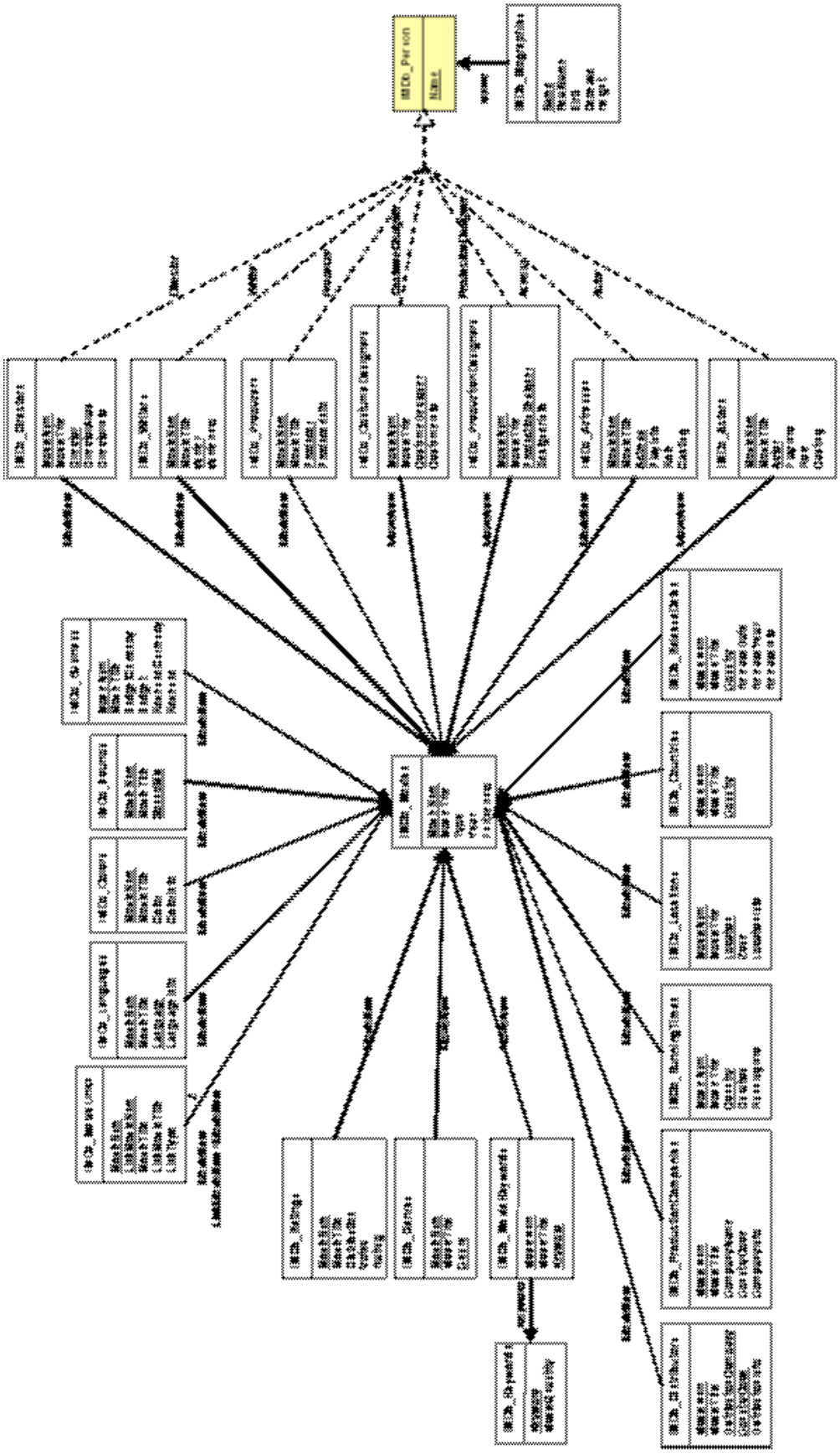
Q7 :
```
SELECT idV, V.prix, V.lieu_depart, V.lieu_arrivee, V.date_depart
FROM VOYAGE V
WHERE V.lieu_depart='Paris' and V.lieu_arrivee='Venise';
```

Q8 :
```
SELECT idV, V.prix, V.lieu_depart, V.lieu_arrivee, V.date_depart, H.region
FROM VOYAGE V, HOTEL H
WHERE V.idH = H.idH and V.lieu_depart='Lyon' and H.region='centre ville';
```

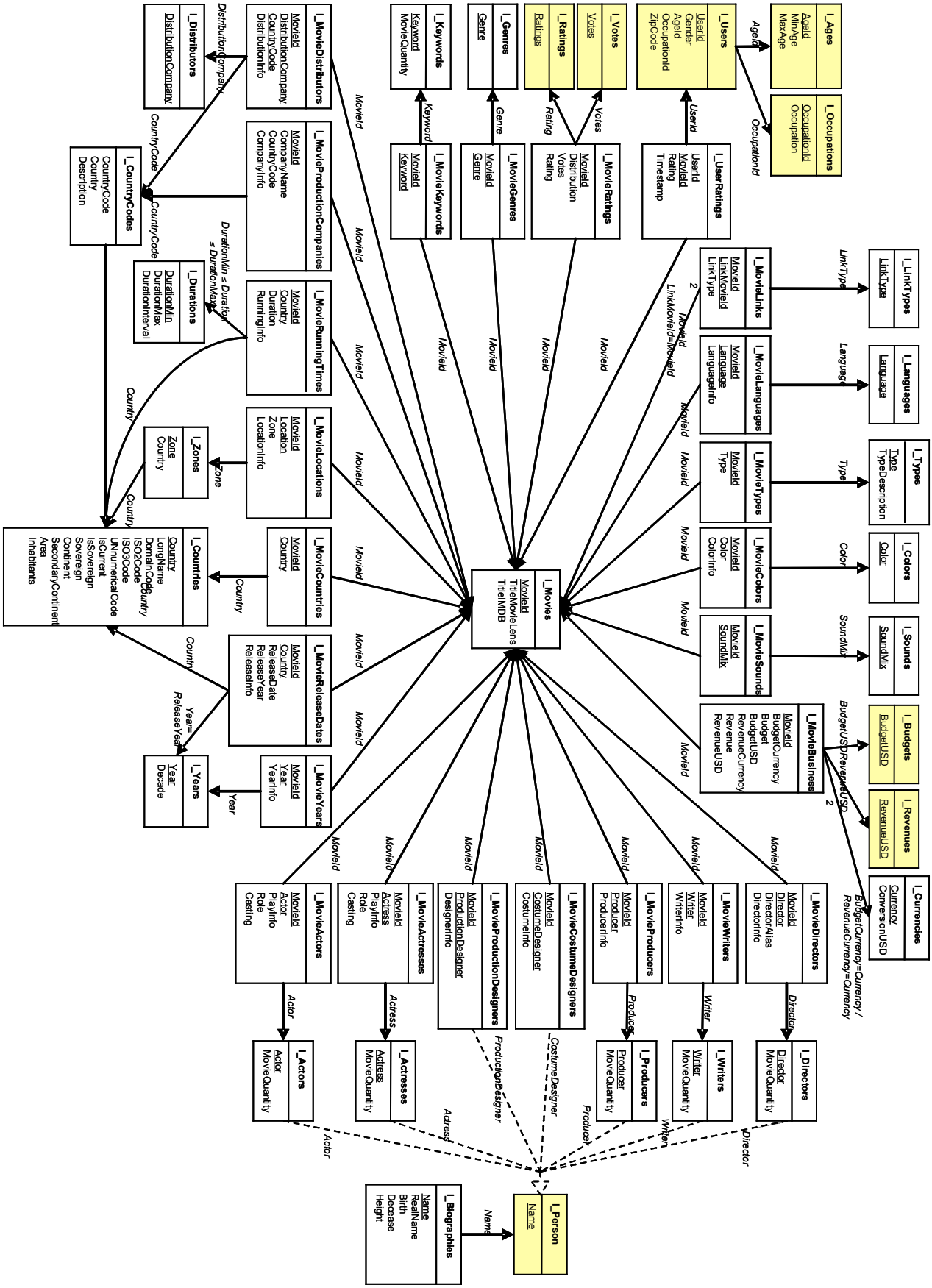
Q9:
```
SELECT idV, V.prix, V.lieu_depart, V.lieu_arrivee, V.date_depart, H.region
FROM VOYAGE V, HOTEL H
WHERE V.idH = H.idH and V.type_formule='week end';
```

Q10:
```
SELECT idV, V.prix, V.lieu_depart, V.lieu_arrivee, V.date_depart, H.region
FROM VOYAGE V, TRANSPORT T, HOTEL H
WHERE V.idT = T.idT and V.idH = H.idH and V.lieu_depart='Paris' and
  T.type_trajet = 'direct' and T.comfort>2 and V.lieu_arrivee='Barcelone' and
  T.moyen='car';
```

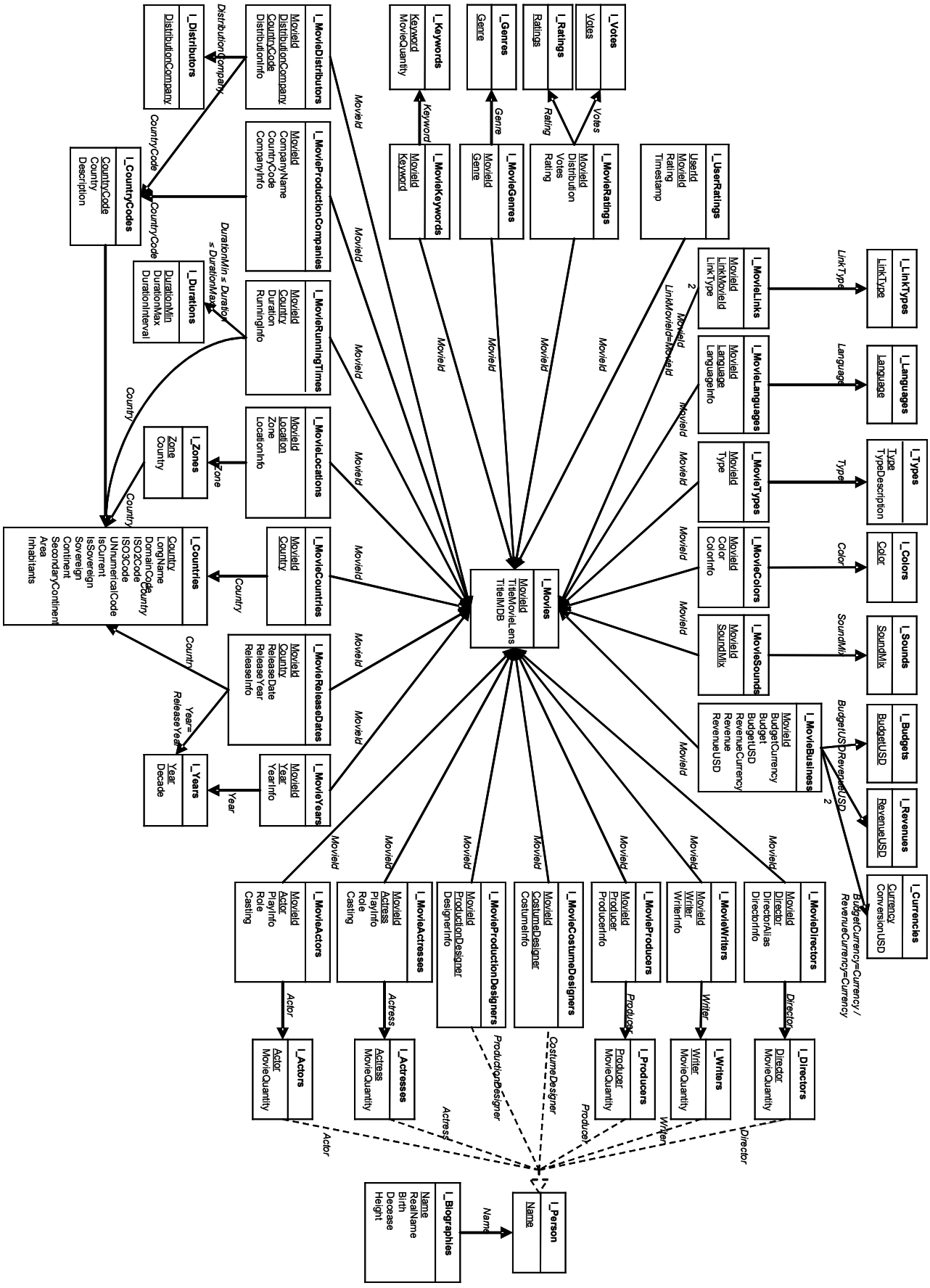
Annexe 2 : Schéma de la base de données qui stocke les données de IMDb



Annexe 3 : Schéma de la base de données intégrée de IMDb et MovieLens



Annexe 4 : Schéma virtuel du benchmark



Annexe 5 : State of the Art

This annexe presents the state of the art on data personalization. The main problem addressed here are: the user profile content, the various techniques of profile construction and profile exploitation.

1. Introduction

This annexe aims to provide a state of the art on user profiling, profile management and profile-based access to information content. This section motivates the importance of user profiles (including their preferences) in selective data dissemination as well as in a personalized access to distributed data sources. The section will also introduce some concepts such as user/community profile, context, query, explicit/implicit preference, user feedback, schema and meta data. Finally, the section will highlight the main goals which drive the writing of the following sections.

Access to relevant information, adapted to the needs and the context of the user is a challenge in an Internet environment (semantic Web), GRID systems and large scaled P2P infrastructures. Many corporate information systems, such as CRM[1] applications, e-commerce systems or video-on-demand broadcasting, require advanced technologies based on their client preferences to make a selective dissemination of their products or their advertisements.

Users of these information systems are surrounded by a proliferation of heterogeneous resources (structured data, textual documents, software components, audio and video data), leading to huge volumes of data. As these volumes as well as the diversify of data increase, information retrieval systems (Web search engines, DBMS, Digital Libraries Systems, etc.) deliver massive results in response to the user requests, thus generating an informational overload in which it is often difficult for users to distinguish relevant information from secondary information or even from the noise. Additionally, new access practices have emerged, hence generating new types of queries such as ranked queries, approximate queries, continuous queries, aggregate and summary queries, context aware queries… Most of these query types intend to fulfil user requirement in terms of implicit or explicit preferences defined over their domains of interest, their spatio-temporal situations, the media used to interact with the systems, etc.

The definition of user profiles (or of community profiles) allows to address this problem through various perspectives such as a personalized information access model, recommender systems, collaborative search engines, publish/subscribe systems. A profile is defined by a set of attributes, possibly organized into abstract entities, whose values can be user-defined or dynamically derived from user behaviour. A profile is supposed to characterize user domain of interest and all his specific features that help the information system to deliver the most relevant data in the right form at the right place and the right moment. These attributes are generally ranked and organized as a preference model which will serve to drive query compilation, query execution and data delivery. Preference expressions may be of several kinds: introducing partial or total orders within data selection predicates or keywords, putting emphasis on some attributes or predicates by assigning relative weights, imposing the subset of predicates that should be fully or partially satisfied (top K predicates), discriminating between query results (top K results), adapting data delivery to the context (interaction media, spatio-temporal position), etc.

[1] Customer Relationship Management

The concept of profile is often designated by the term context. However, this term is often limited to the description of user or system environment. Some context definitions focus on the geographical location of the user, others include the media used to interact with the system. The concept of profile may also overlap the notion of query. In some applications such as publish/subscribe systems, a profile includes all what the user needs in terms of data. Other applications make a clear distinction between profiles and queries: a profile is a user model which specifies the user domain of interest and the most general preferences that distinguish this user from the others, while a query is an on-demand user need which is evaluated with respect to a certain profile. All queries issued by the same user are evaluated with respect to his specific profile. The same query issued by different users may have different results as it is evaluated using different profiles. Profiles and queries should be considered as orthogonal concepts, although in some information retrieval systems both concepts are superimposed.

The goal of this annexe is to provide an extensive state of the art on user profiling, profile management and profile-based data delivery. Section 2 will summarize the main application domains where personalization techniques offer the most effective results. The section ends with a set of main features characterizing the different types of personalization systems. In section 3, we summarize the main profile representation models proposed in the literature. Section 4 surveys the fundamental techniques used to build user profiles, using in particular the logs of their past transactions. Section 5 synthesises the main techniques used to exploit user profiles and preferences. Finally, section 6 concludes the annexe.

2. Application domains and types of personalization systems

Data personalization has been addressed in different domains such as information retrieval, database and human-computer interaction. In information retrieval systems, the user is fully involved in the query evaluation which is conducted as a stepwise refinement process where the user can decide at each step which data he likes and which data he dislikes. The personalization is then considered as a machine learning process based on user feedback. In database systems, it is not usual to involve the user in the data retrieval process: a database query contains generally all the necessary criteria to a selection of relevant data. Data personalization is thus considered under the viewpoint of query language extensions (generally SQL) and query reformulation process using user profiles. In human-computer interaction, user profiles generally define user expertise with respect to the application domain in order to provide them with appropriate interfaces and dialogues. Data delivery modalities, graphical metaphors and level of expertise of the dialogue constitute the main personalization issues in this context.

This section presents some examples of personalized applications and discusses the main features of a personalized system.

2.1 Examples of personalized applications

There exist lot of applications which offer personalized services. This section provides examples of such applications which aim to give an overview of what personalization could be.

PANDORA

A first example of personalized application is PANDORA[6]. It is a service that enables customers to enjoy songs they prefer and help them discover new music *they will love*. In the

[6] http://www.pandora.com

heart of PANDORA we find *the Music Genome Project*[7] which analyses the music in order to capture the complex musical DNA of songs. This is done with the help of a large team of highly-trained musicians. This team ended up identifying hundreds of attributes "genes" that characterize songs. Taken together, as for humans, these genes capture the unique and magical musical identity of a song. These genes concern everything from melody, harmony and rhythm, to instrumentation, orchestration, arrangement, lyrics, and of course the rich world of singing and vocal harmony.

The use of PANDORA is very easy and intuitive. Users have to subscribe in order to get an account. In the subscription step, each user is asked to give his age and gender. After registration, the user is invited to create his first radio station by entering a "song title" or an "artist name". Song title will give very specific information about the musical elements that the user wants to hear. It is advised to use more than one song in order to provide some variety. When listening, the user cans "*Thumbs Up*" or "*Thumbs Down*" the song (feedback). A thumbs up vote has several effects, (i) the very next set of songs for this station will be directly generated using the song which received the thumbs-up vote. It's a great, because it's real time. (ii) Future sets of songs are more likely to have similar musical traits. Finally, all songs that have received a thumbs-up vote on this station will become members of a new invisible starting point. On another hand, thumbs down vote will immediately stop a song which is deleted permanently from this station's playlist. An artist whom two songs are thumbs down, will be banned from playing on this station. Finally, future sets of songs are less likely to have similar musical traits. Note that PANDORA music licenses don't allow users to replay or rewind a specific song, or to play a particular song or artist on demand.

TiVo

TiVoBox[8] is a new US phenomenon. It is an engine that automatically finds and digitally records all of viewers' favourite shows every time there are on. TiVo is not a simple DVR (Digital Video Recorder), in fact, it allow services such as "*Finding Great Entertainment*" which makes easy for viewers to find what they want and collect shows by responding to viewer interests. This is achieved thanks to personalization and it is done through two mechanisms. The first one is Universal Swivel® Search which instantly finds programs the viewer is looking for by introducing information such as: actor names, show titles, or any other word/sentence that may express viewer interest. The interface used to do this is shown in Figure A5.1. Swivel Search creates a "*Wishlist Searches*" that will be updated over time with new programs that match well with viewer's interests. When the viewer is watching TV, he may automatically be swivelled from a program to another with regard to these lists.

[7] http://www.pandora.com/mgp.shtml
[8] http://www.tivo.com

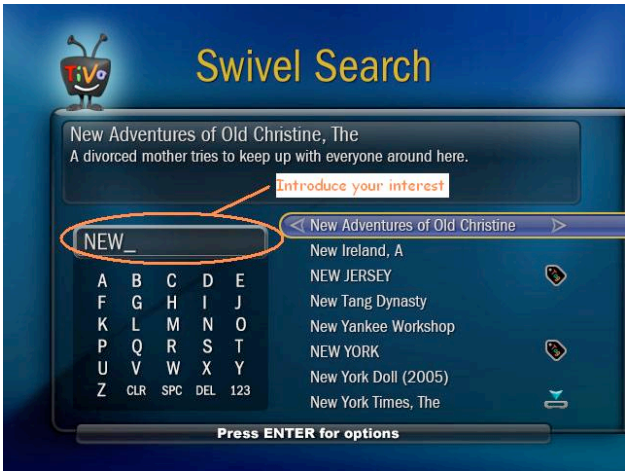


Figure A5.1 : TiVo Universal Swivel Search Engine

The second personalized service offered by TiVo is "*TiVo Suggestions*". Viewers can rate any program they watch whether it is live, recorded or listed in the program guide. Rating is done through special buttons "Thumbs up" and "Thumbs down" on "TiVo remote control" as shown in Figure A5.2. Viewers can give a program up to three Thumbs Up "*great!*" or three Thumbs Down *"terrible!"* TiVo Suggestions uses these ratings to create a list of programs viewer might like. The more shows viewer rates over time, the better TiVo Suggestions will get at finding the right programs for him. Any time a viewer record a program, it automatically gets one Thumbs up.

Figure A5.2 : Part of TiVo remote control

A major problem for such DVR is privacy since TiVo records usage data for their own research and they also sell it to other corporations such as advertisers. In fact, in The New York Times article of July 26, 2006 it is reported that TiVo is starting a research division to sell data about how its 4.4 million users watch commercials or, more often, skip them. And this may uneasy some viewers.

MyNews

Another personalization project is the *MyNews* prototype [Dem02] which allows users to watch what they want, when they want. *MyNews* is based on the Real Time Television Content Platform (RTTCP) technology. RTTCP is a device that includes PVR (Personal Video Recorder). This component combines inexpensive computing and storage in a consumer device that is capable of recording broadcast content according to viewers' preferences. *MyNews* reposes on PVR Agent technology. The agent stores viewer preference

data, it then watches the broadcast news from the news sources. Agent can operate in two modes; it can record all the news and sort them according to preferences. It can also record only news that fall within a certain threshold. A crucial problem that occurs in such personalized application is to find the best way to get personal preferences from the user. This involves striking a balance between relatively *broad and deep taxonomy* (ontology of news domain) and the need to create *an easily navigable interface* (TV, PC, PDA, Mobile Phone…etc.). The development of fairly broad and detailed taxonomy is an ongoing problem for most content providers. However *forcing users to input preferences based on full taxonomy would be overwhelming* for typical user. For these reason the *MyNews* preference interface presents only a small representative sample of the taxonomy to the user. This is done thanks to sparse interface that represents news taxonomy in term of *categories*, *peoples* and *location*.

User is invited to rate items on a -2 to +2 scale, not rated items are assumed to have a neutral value (0). In the case of deeper hierarchy item, all child items are given the same rating. This constitutes the initial user profile. Two techniques are then used to build richer profiles:

- Inferring demographics data. For example a user who is interested in *finance* and *Bill Gates is seemed to be interested in American economic.*
- Monitoring user interactions. Assume that user has rated +2 sport categories; obviously Basketball will receive (initially) the same score. If user changes the channel every time that the show is related to Basketball, the system must be able to infer that this user hates basketball sport, and must decrease its score.

In order to allow matching preferences to content, the content is tagged using the same taxonomy on which users have expressed their preferences. Currently, *MyNews* prototype uses similarity measures that are based on vector space models (see section) to match content to user's preference profile.

JobFinder

Another example of application is JobFinder[9] which uses the work carried out in the CASPER project (Case-Based Profiling for Electronic Recruitment) [BRS00, RBS00]. In JobFinder the result selection is processed on the server side and personalization is achieved on the client side. This is done in order to protect personal data. Each job advert is described by a fixed set of features which can be used in user queries. Users submit queries by giving values to a subset of job features (ex. minimum experience, salary etc.). Then a server side search engine selects adverts which are similar to the query criteria. Servers use similarity-based techniques instead of exact match techniques which enable enlarging the search space. Consequently a job may appear in the result even if its values don't match exactly those in the query. When the results are sent to the client-side component, a personalization step is applied before presenting them to the user. This step consists in classifying jobs in two categories: relevant and not relevant. This classification is performed using previously viewed jobs. Each new job is compared to the known ones and its relevance is estimated with respect to the relevancies of the most similar known jobs. In this manner, only relevant elements are returned to the user.

Apt Decision

In JobFinder, the user is not involved in the information retrieval process. Results are delivered without giving the user the opportunity to react. An example of application which

[9] http://www.jobfinder.ie

uses the interactions between the user and the information system in order to achieve personalization is described in [SL01]. This is an application for house rental which uses an agent, named Apt Decision, to learns the users interest. An interesting contribution of Apt Decision is that it gives to the user the opportunity to react not only to apartment offering, but also to a particular feature of the apartment. The user interest is captured in a profile composed by 12 feature slots: 6 slots for positive features and 6 slots for negative ones. Two scenarios are possible: (i) the user acts manually on his profile or (ii) the system updates the slots automatically. In the first case, the user drags manually features into the slots. Once the slots have been modified, a new research can be performed with the new features. In the second case, at each iteration the user is given two examples of apartments and he chooses one of them. The characteristics of the chosen apartment are then analyzed in order to update the slots contents. This process can be repeated until the user is satisfied.

Quickstep

Quickstep is a recommender system, described in [MDS01], which address the real-world problem of recommending on-line research papers to researchers. User browsing behaviour is unobtrusively monitored via a proxy server, logging each URL browsed during normal work activity. A Nearest Neighbour Algorithm classifies browsed URL's based on a training set of labelled example papers and stores each new paper in a central database. The database of known papers grows over time, building a shared pool of knowledge. Explicit feedback and browsed URL's form the basis of the profile for each user. Every day a set of recommendations is computed using the correlations between the user profile and the topics of the classified papers. All actions of the user on these recommendations are stored to form the user feedback. Users have the possibility to provide new examples of topics and correct paper classifications when they are wrong. In this way, the classification accuracy improves over time. In Quickstep user profiles are based on research paper topics ontology. This allows inferences from the ontology to assist profile generation. Topic inheritance is used to infer interest in super-classes of specific topics. Sharing interest profiles with a specific ontology is not difficult since they are explicitly represented using ontological terms.

Amazon.com

The famous e-commerce bookstore application of Amazon.com [SKR01] is structured with an information page for each book, giving details of the text and purchase information. The Customers Who Bought feature is found on the information page for each book in their catalog. It is in fact two separate recommendation lists.

Amazon.com is an e-commerce bookstore application [SKR01]. Each book is associated with an information page which gives details about the book's content and purchase information. Amazon.com proposes recommendations via the "Customers Who Bought" feature which is available on each information page in their catalog. This features represents in fact two separate recommendation lists. The first one recommends books frequently purchased by customers who purchased the selected book. The second one recommends authors whose books are frequently purchased by customers having purchased books of the selected book's author. Amazon encourages direct feedback from customers on the books they have read. Customers rate books they have read on a 5-point scale from "hated it" to "loved it." After rating a sample of books, customers may request recommendations for books that they might like. At that point, a half dozen non-rated texts are presented according to the user's tastes.

Amazon enables customers to be notified via email when new items which match their interest criteria are added to the catalog. This is done through the "Eyes" feature. Customers enter queries by giving information about the book's they are interested in. Features which

can be queried include the author, the title, the subject, the ISBN, and the publication date. Customers can use both simple and complex Boolean-based criteria (AND/OR) for notification queries. One of the interesting variations of the Eyes system allows queries to be directly entered from any search results screen, creating a persistent query based on the search.

eBay.com

eBay.com is another example of personalized e-commerce application. It uses a *Feedback Profile* component which enables buyers and sellers to contribute to the construction of the feedback profiles of other customers [SKR01]. The feedback consists in giving satisfaction rating (satisfied/neutral/dissatisfied) and comments about the other customers. Feedback is used to provide a recommender system for purchasers by giving them the possibility to view the profile of sellers. This profile consists in a table which contains on one hand the number of rating of each type that the user has received during the last 7 days, the last month, and the last 6 months, and on the other hand an overall summary of the comments. This way, customers can browse the individual ratings and comments for the sellers.

Another feature of eBay is the personal shopper. It allows customers to indicate items they want to purchase. This is done by giving search criteria, which includes the price limit, in the form of set of keywords. Those set of keywords form the customer's query. Periodically (one or three day intervals) the site executes the query over all auctions of the site and sends the customer an email with the obtained results.

Examples of personalized applications show that there are as many personalisation approaches as existing applications. The difference between those applications is as well in the kind of data they store about users as in the manner this data is obtained and used.

Personalized applications may produce recommendations at varying degrees of personalization. The degree of personalization encompasses several factors including the accuracy and the usefulness of recommendations. Accuracy measures how correct the system is while usefulness includes factors such as serendipity and individualization (the system provides different recommendations to different people). [SKR01] identifies three common levels of personalization:

- ✓ *None personalized*: Recommender applications provide the same recommendations to each customer. Good examples of none-personalized recommendations are the *best seller* books and top ten *box office* movies.
- ✓ *Ephemeral personalization*: Recommendations are based only on the current customer's inputs. This is a step above non-personalized recommenders because it provides recommendations that are responsive to the customer's navigation and selection.
- ✓ *Persistent personalization*: It is the most highly-personalized recommender applications. It recommends different items to two different customers even if they have visited the same items. This kind of recommendation requires systems to maintain persistent data about users.

2.2 Main features of a personalization system

There are many features or modalities which characterize a personalized system. Among these modalities, we can mention the following ones:

- *Matching modality (content vs. collaborative)*: this modality specifies what kinds of objects are considered during the matching performed by the information retrieval

system. When the matching concerns in one hand a user profile (or a user query) and in another hand a data resource, it is called a content-based approach. Content-based systems exploit description (semantic) of items to extract more relevant results. When the matching considers different user profiles, it is called a collaborative approach. Collaborative systems exploit ratings of other users (through their profiles) to answer the query of a given user.

- *Interaction modality (pull vs. push)*: this modality specifies whether the interaction between the user and the system is pull or push. Pull interaction defines systems where queries are issued by the user and are matched either against the information system content (content-based filtering) or against other user profiles (collaborative filtering). Push interaction defines systems where user queries (profiles) are defined once for all as subscriptions against which new resource events are matched. A resource event may be, for example, a new tuple in a database, a new service or a new video content. Pull systems typically behave as database systems and Google, while push systems behave as publish/subscribe systems and specialized Internet portals.
- *Delivery modality (selective filtering vs. ranking)*: this modality defines which set of data is delivered to the user. Selective filtering systems deliver only relevant data, that is data whose semantic distance with the query (or the profile) is bounded by a threshold, while ranking systems deliver all matched results ordered by their semantic distance.

These three modalities are orthogonal; the same system can be characterized with one or several of them. However, these modalities are not sufficient to fully describe a personalized system. Actually, there is no common architecture for personalized systems. As seen before, each domain of application has its own architecture for personalization. Nevertheless, a rough picture can highlight the main components of a personalized system (Figure A5.3). Among these components, we can mention the followings:

- An information retrieval system which can be characterized by one or several of the previous modalities depending of the application type;
- A set of sources which might be data sources or software services. They are described by a set of meta data;
- A set of user profiles, each profile describing the domain of interest and preferences of the corresponding user;
- A set of interaction contexts, each context describing preferences about user-system interaction.

Profiles and interaction contexts constitute a user knowledge model which will be exploited by the information retrieval system either to answer queries or to disseminate information (or to recommend products). User preferences, whatever they concern system adaptation or content delivery, are the main knowledge which characterizes a personalization system whose target is to augment the user satisfaction.

Next section will describe in detail the definition and representation of user profiles and preferences, as well as their exploitation in different research prototypes.

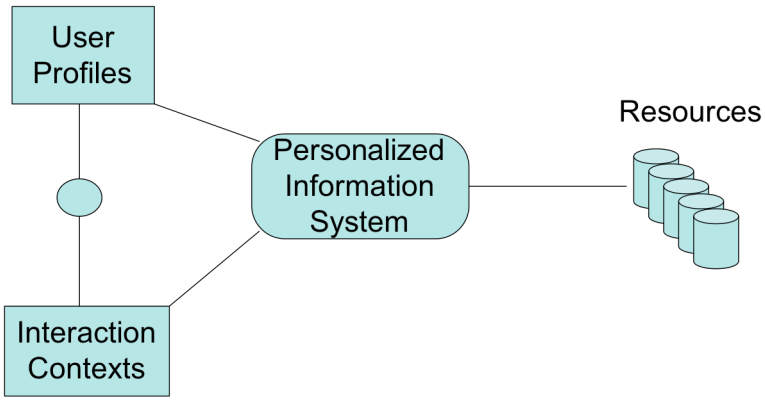


Figure A5.3 : Main components of a personalization system

3. Definition and representation of user profiles

This section reports on the various definitions and representations of the knowledge composing user profiles. It summarizes different taxonomies of profile attributes and preferences which have been proposed in the research projects and in commercial products. Through various profile examples, we will describe formal models used to represent these profiles.

3.1 Examples of user profiles

A very simple kind of user profile contains examples of documents associated with their relevance. An example of such user profile, as defined in the CASPER project [RBS00, BRS00], is given in Figure A5.4. It shows a typical profile derived by CASPER for a user interested in job announcements. A user profile contains the job announcements consulted by the user and information about the actions that the user has performed on them. The features describing actions include the action type, the number of clicks and the time spent to do it. An in-depth study of implicit measures of user interest and preference can be found in [DJBW03, KT03]. Based on this derived profile, the information retrieval system uses a set of decision rules to filter jobs in further queries. In the example of Figure A5.4, job56 and job45 are considered relevant to the user regarding the action types made on these jobs (apply to the job and email the job announcement to a friend) while job5 is less important as it was only read.

Job Type	Action Type	Number of Clicks	Reading Time
job5	Read	1	234
job56	Apply	2	186
job45	email to a friend	1	54

Figure A5.4: Example of user profile in the CASPER project

These two relevant announcements constitute interesting prototypes that will be used in a case-based reasoning approach to determine whether other new announcements are relevant or not for this user.

Another example of user profile can be expressed using utility functions over a domain of interest [CGFZ03]. Two complementary clauses, DOMAIN and UTILITY, enable to define respectively the domain of interest of the user as a set of abstract subjects, and the relative importance of these subjects as a set of equations. The example of user profile of Figure A5.5, taken from [CGFZ03], defines a traveller profile whose domain of interest is composed of three subjects: car rental companies (CR), airport shuttles schedules (ASS) and availability of city maps (CM). Three utility equations complement this definition by setting

the relative importance of these subjects as well as their dependencies. The first equation means that the first 2 shuttles schedules have a utility equal to 5 while the remaining schedules have a utility equal to 1. The second equation means that the utility of the first two city maps objects is 3, and the remaining objects of that category have a utility of 0. The third equation defines the context where the car rental utility function has a meaning, that is when there is at least one city map. In that case, the first 3 companies have utility 2 while the others have utility 0.

```
PROFILE Traveller
   DOMAIN
        CR = www.hertz.com (car rental companies)
        ASS = « Airport shuttle schedules»
        CM = « city maps and maps from Airport to the city»
   UTILITY
        U(ASS) = UPTO (2, 5, 1)
        U(CM) = UPTO(2, 3, 0)
        U(CR [#CM > =1]) = UPTO (3, 2, 0)
END
```

Figure A5.5: Example of utility functions based user profile [CGFZ03]

Such profile is used in cache management. The main idea is to maximize the utility of the cache, which is defined as the sum of the utilities of the objects in the cache. Suppose that objects are Web pages and the cache contains 4 pages of car rental companies (CR), 1 page with city maps and maps from Airport to the city (CM) and 3 pages with airport shuttle schedules (ASS). The utility of this cache is the sum of utilities of contained pages. The utility of the CR pages is given by the third utility function. First, there should be at least 1 CM page (#CM > =1). If this condition is satisfied, the utility of each CR page will be considered. The 3 first CR pages have a utility of 2 while the remaining one has utility 0 (UPTO (3,2,0)). The utility of the 4 CR pages is 2+2+2+0 = 6. Using the same process, the utility of the CM page is 3 and the utility of the 3 SS pages is 5+5+1=11. The global cache utility is then 6+3+11=20.

In [KI04a] the user profile is defined with respect to a database schema as a list of weighted predicates which represent the most frequent selection predicates that are supposed to appear in user queries. The user interest on each predicate is represented by a weight between 0 and 1 to emphasize the relative importance of one predicate with respect to others.

Consider the following database schema:

```
TRANSPORT(idT, mean)
HOTEL(idH, nbStars, region)
TRAVEL(idV, price, departure, arrival, nbDays, idH, idT)
DEPARTURE (idD, idV, date, hour)
```

Figure A5.6 presents the profile P_u of a user who lives in *Paris*, likes *two days* trips in hotels situated in the *centre city* and prefers travelling by *train* than by *car*.

This profile is further used to enrich user queries by introducing new selection predicates taken from the profile and improving the relevance of the query.

P_u =		
{TRAVEL.idT = TRANSPORT.idT	1	(a)
HOTEL.idH = TRAVEL.idH	0,9	(b)
TRAVEL.departure = 'Paris'	0,9	(c)
HOTEL.region = 'centre city'	0.8	(d)
TRAVEL.nbDays = 2	0.4	(e)
TRANSPORT.mean = 'train'	0.3	(f)
TRANSPORT.mean = 'car'	0.2 }	(g)

Figure A5.6: Example of weighted predicates based user profile

The examples described above show that the user profile definition may vary according to application domains and technologies. However, regardless to a specific application, a profile defines at least a domain of interest (user's interest with respect to a certain business perspective), and a set of preferences over this domain. We will show in the next section that a user profile may contain much more knowledge, such as personal data [Pot04], data delivery preferences [GM05], data quality [BR02] and security preferences [CLM02]. The next section defines the main categories of knowledge composing a user profile.

3.2 Types of user preference

This section lists the possible features of a preference [Cho02, SPT06, SHY04, VFC+06, BHL01]. A preference can be described using the following characteristics:

Constraint type. According to the constraint type a preference can be *hard* or *soft*. Hard preferences represent mandatory constraints and have to be always satisfied, while soft preferences have to be satisfied if possible. In other words, if for a given soft preference there are not any results, the preference is not considered while if the same situation occurs for a hard preference, the user is delivered an empty result set.

Cardinality. The cardinality of a given preference shows if it is single or composite. With respect to its cardinality, a preference can be *atomic* or *multiple*. An atomic preference is a simple preference expression while a multiple preference is a composition of atomic preferences.

Composition type. The composition type is a feature which only refers to multiple preferences. It defines the relative importance of the atomic preferences in the composition. Atomic preferences can be combined in *independent* or *prioritized* way. Independent preferences are considered equally important. When evaluating independent preferences, it is not important which preferences are satisfied but their number. On the contrary, if the order in which preferences are satisfied is important, they are combined in prioritized manner.

Metric type. Preferences can be expressed using a *qualitative* or a *quantitative* approach. Quantitative preferences are expressed indirectly using scoring functions. These functions assign a numeric score to each element thus resulting in elements ordering. Qualitative preferences are expressed usually directly using binary relations. They are more general than quantitative ones because the order resulting of a scoring function can be expressed with binary relations, while not every plausible preference can be captured using scoring functions.

Point of view. With respect to the point of view, preferences can be *intrinsic* or *extrinsic*. Intrinsic preferences are expressed on the content of the elements in which the user is interest in, while extrinsic preferences refer to external characteristics related to the information container or the production process.

Persistency. The persistency of a preference shows if it is *persistent* or *ephemeral*. Persistent preferences are relatively stable and are used in multiple information research sessions. In contrast, ephemeral preferences are only considered during one session and disappear after it ends.

Situational utility. The situational utility is only defined when the preferences have to be adapted to a given querying situation. In this case the preferences which have to be considered according to the situation are *relevant* and all other preferences are *noisy*.

Valence. The valence of a preference is relative to the value for which the preference is expressed. It can be positive, negative or indifferent. A positive valence expresses value liking, a negative one disliking and a valence equal to 0 indicates indifference.

Concern. The concern of a preference shows if the user likes the presence or the absence of the value considered in the preference.

Elasticity. According to the elasticity criterion, preferences can be *exact* or *elastic*. The elasticity of a preference depends on the nature of the values on which it is expressed. For categorical discrete values, the preference can be satisfied or not at all. In this case the preference is exact. For continuous numeric values, the preference can be partially satisfied and is considered to be elastic.

To those existing preference categories can be added the preference **semantics** and the **evaluation mode**. The semantics of a preference shows how to identify the relevant elements. There are three main types of preferences according to their semantics:

- *Top N and Best preference*: This kind of preferences can be only expressed on ordered sets of elements. Their evaluation depends on the order type. For total orders the Best-type preference selects only one element while for partial orders only non-dominated elements are chosen. The Top N preference behaves almost like the Best one but imposes the number of results to N elements. If the number of the best elements is greater than N, options can be defined to select randomly the N ones or to take the first N ones among these. If the number of best elements is less than N, the top-N operator takes the remaining elements among the ones ordered immediately after the best elements.
- *First K preference*: This type of preference is used to limit the search to a subset of possible results, regardless to any selection criteria, except the order on which the elements are computed or delivered. Two situations may occur based on the way of results computing: one at a time or all at the same time. In the first case, where the results are computed one after another, the first K ones are returned to the user according to the moment of their arrival. In the second case all results are evaluated at the same time and K among them are randomly selected.
- *Near X preference*: For this type of preference, there must be a set of relevant examples of elements. The main idea is to find other elements which are similar to the known ones. This task is usually performed using distance functions. They assign a distance score to each pair formed by taking one new element and one known element. According to these distances, the relevance of the new elements can be estimated.

Finally, the evaluation mode of a preference gives information about how the preference is considered by the information system during the query evaluation process. There are two possible evaluation modes:

- *Global evaluation*: The global evaluation mode consists in taking into account the whole set of preferences at the same time. The results considered to be

relevant are returned to the user without giving him the possibility to ask for more answers. If new results have to be computed, the user has to resubmit a new query.

- *Interactive evaluation*: In interactive evaluation, the user and the information system work together for finding the relevant elements. There are two interaction modes: with or without changing the research criteria. In the first case, the user is given intermediate results on which he can express a satisfaction level. According to this information, the system refines the research criteria and repeats the research process until the user is satisfied. In the second case, the research criteria remain the same and the most relevant elements not yet retuned are given to the user.

This section identified the categories of preferences. It also gave a brief description for the possible values of each preference feature. The next section shows several examples of preference expression formalisms and discusses the categories on which they belong.

3.3 User Profile and Preferences Formalisms

This section presents the formalisms, which can be used in order to describe the user profiles and preferences. In fact, in many cases, user profile is reduced to user preferences. This is done through several examples of user profiles or preference representation formalisms. Most of those formalisms refer to the domain of interest of the user and enable expressing soft, atomic, intrinsic and persistent preferences. Consequently, only special cases, where the formalisms enable expressing preferences of other types or/and preferences on different profile categories, will be pointed in this section. On the other hand, we will try to present different user profile representations that are mentioned in literature from the most basic keyword representation to the complex multidimensional one.

3.3.1 Weighted Keyword Profile

The profile content varies with technologies and application domains. In Information Retrieval (IR), documents and user profiles are usually represented by a set of possibly weighted keywords [CN04]. The weight of a keyword corresponds to the level at which it is relevant for the document representation or to the level of user interest. The weights of the keywords representing documents are then compared to those in the user profile in order to identify the most relevant documents. The comparison between two sets of keywords is based on distance functions and only documents, which are close to the user profile, are selected.

Key words**: (data, user, software, car, techniques, model, colour, price)**

d = (0,4; 0,5; 0,2; 0,0; 0,1; 0,6; 0,0; 0,0)
P_u = (0,0; 0,7; 0,5; 0,0; 0,0; 0,4; 0,0; 0,0)

Figure A5.7: Example of keywords based profile representation

Figure A5.7 shows a document (d) and a user profile (P_u) which are represented by the weights of a fixed set of keywords. The weights in the document vector express the level at which the terms are representative for the documents while weights in the user profile vector express preference levels. This approach uses intrinsic, quantitative preferences, where the user is interested in the presence of the terms composing his profile into the documents.

The limit of the weighted keywords representation formalisms is that they consider the terms in isolation. In some situations, the semantics of a term can be ambiguous (ex. JAVA can be a programming language, a coffee or the name of an island), thus compromising the relevance of the retrieved documents. In order to better capture the semantics of the terms which are of some interest for the user, some approaches propose to group terms in sequences [ES95]. Sequences are words which appear together in documents. They are represented by a directed graph where nodes are terms and edges correspond to sequences. Each edge has two weights: one representing the probability that the two terms appear together and the second corresponds to the relative importance of the sequence according to other sequences in the graph. A pair of terms is called S-entity. S-entities can be grouped in Supervisors to form sequences of more than two terms.

3.3.2 Formula-based Profile

In database field, user preferences are expressions which enable to order the information elements [BKS01], [Cho02], [CT02], [Kie02]. An example of such approach is the WINNOW operator which uses preference formulas [Cho02]. A preference formula is a logic expression which defines tuples over a database schema and orders them according to the values of their attributes. Figure A5.8 shows a user preference (C) for low cost travels. The preference for a tuple of the table TRAVEL (t_1) over another one (t_2) according to the preference formula C is denoted by $t_1 \succ_C t_2$. It occurs if t_1 has the same values for departure, arrival and number of days as t_2 and if its price is lower.

View: TRAVEL(price, departure, arrival, nbDays, hotel, transport)

Preference for low cost travels:

C = (P, D, A, ND, H, T) > (P', D', A', ND', H', T') ≡

(D=D' ∧ A = A' ∧ ND = ND' ∧ P<P')

Figure A5.8: Example of preference formula

This formula is then used by the WINNOW operator for selecting not dominated tuples. In other words, each pair of tuples t_i, t_j, returned in the result of the WINNOW operator, must have the same values for the preferred attributes or have to be indifferent (neither $t_i \succ_C t_j$ or $t_j \succ_C t_i$ holds). In this approach it is possible to apply the WINNOW operator more than once. The result of each iteration is composed by the not dominated tuples not yet returned. This approach enables expressing atomic, qualitative, intrinsic preferences which are evaluated iteratively without changing the search criteria.

Preference operators

An example of approach where it is possible to define multiple preferences is presented in [Kie02] and [Kie05]. Preferences are expressed for attributes appearing in some relation schema. Each atomic preference is of the form:

base *op_name(A, parameters) {definition of* $<_p$*};*

where ***base*** is a keyword of the language, *op_name* is the new operator's name, *A* is the attribute of a relation R on which the preference is expressed, *parameters* is the set of parameters necessary for the preference definition and *definition* is the logic expression defining the preference operation.

Using this syntax, preference operators are proposed. They are defined for attributes which can take numeric or alphanumeric values. Examples of preferences for numeric attributes are: AROUND, BETWEEN, LOWEST, HIGHEST and SCORE, while for alphanumeric attributes there can be POS, NEG, POS/POS, POS/NEG, EXPLICIT operators. Consider for example an attribute A of the schema of a relation R, which can take numeric values from the domain dom(A). Let x and y ∈ R be two tuples from R. The LOWEST operator expresses preference for the lowest values of the attribute A. It can be defined as follows: LOWEST(A) {$x <_P y$ iff $x.A > y.A$};

In the same way, if A is an attribute taking alphanumeric values, the POS preference operator can be defined with the expression:

POS(A, POS-set{V_1,…, V_m}) { $x <_P y$ iff $x.A \notin$ POS-set $\wedge$ $y.A \in$ POS-set }; where V_1,…, $V_m \in$ dom(A). It expresses preference for elements which values for A are in the positive set.

The atomic preferences can be combined in order to form complex preferences. Preference composition operators are defined using the following syntax:

complex $Pref_1$ op_name $Pref_2$ {operator definition $<_p$}

There are two main preference composition operators: one for equally important preferences (denoted with "⊗") and one for hierarchic preference composition (denoted with "&"). There definitions can be given with the expressions:

Let A, B ∈ R and $(A_1, B_1), (A_2, B_2) \in dom(A) \times dom(B)$.

complex $Pref_1 \otimes Pref_2$ $\{(A_1,B_1) <_P (A_2,B_2)$ iff
$(A_1 <_{P1} A_2 \wedge (B_1 <_{P2} B_2 \vee B_1 = B_2)) \vee$
$(B_1 <_{P2} B_2 \wedge (A_1 <_{P1} A_2 \vee A_1 = A_2))\}$;

complex $Pref_1$ & $Pref_2$ $\{(A_1,B_1) <_P (A_2,B_2)$ iff
$A_1 <_{P1} A_2 \vee$
$(A_1 = A_2 \wedge B_1 <_{P2} B_2)\}$;

Consider for example the view TRAVEL in Figure A5.8 and consider two atomic preferences: one for lowest price travels (P_1) and one for travels which departure is 'Paris' or 'Versailles' (P_2). They can be expressed by:

P_1=LOWEST(price);

P_2=POS(departure, {'Paris', 'Versailles'}).

If the user considers that the price is more important than the departure's city, he can define a hierarchic preference (P_3):

$P_3=P_1\&P_2$;

This approach enables expressing both atomic and multiple preferences. Multiple preferences can be independent or prioritized and the user can be concerned by the presence (POS) or the absence (NEG) of the values.

Utility functions

In some situations the number of results is more important than their content. This is the case of the user profiles based on utility functions over a domain of interest presented in Figure A5.5 of section 3.1 [CGFZ03]. As mentioned before, the utility of elements for an object *E* of the domain of interest, can be expressed using function of the form:

U(*E*[*condition*])=UPTO(I, J, K) where *condition* is optional expression which shows how the utility of *E* is affected by the presence of elements from the other types of objects. UPTO is an operator indicating that the first I elements of *E* have an utility of J, and all other elements an utility of K. Utility functions express First N, quantitative, extrinsic preferences over the domain of interest.

Horn clauses preferences

In some information systems, preferences are expressed using Horn clauses like formalisms [KI04b, APV02, GL94]. In [GL94], concepts are defined with Horn clauses and user preference is expressed of the form:

predicate: degree of satisfaction [← condition], where degrees of satisfaction are previously defined and ordered by the user.

This formalism is used to define intrinsic qualitative preferences over the Domain of Interest. The user profile in Figure A5.9 defines 4 satisfaction levels (from very good to very bad). There is only one rule expression user dislike for flights with stopover in Paris.

Degrees of satisfaction:	very good >> good >> bad >> very bad
Rules:	stopover(Flight, Airport) : bad ← airport_paris(Airport)

Figure A5.9: User profile as in [GL94]

Another example of Horn clause-based profile is presented in [APV02]. Clauses define concepts, which are ordered using rules. The user profile in Figure A5.10 defines two concepts: action films (N_1) and centre city region (N_2). The rule R expresses user preference for watching action films with respect to going to cinemas in the centre city on weekend. In other words, in weekends the satisfaction of N_1 is more important than this of N_2. Using this formalism, the contextual utility and the composition type of the preferences can be captured.

N1: action ← not thriller, not comedy
N2: centre city ← not suburb, not peripheral

R: N1>N2 ← Weekend

Figure A5.10: User profile as in [APV02]

Rule based preferences

Finally, the user profile in [KI04b] enables adding predicates to the user query using rules. Each rule is associated with a weight in [0,1] which expresses its relative importance according to other rules. The user profile in Figure A5.11 defines 3 rules. The first one sets the departure city to Paris for a query made on the relation TRAVEL, the second one sets the number of days of a travel to 2 and the last one expands a query with the relation TRANSPORTS and sets the mean to train. Consider a user query asking for travel which cost 1000 euro: TRAVEL(_,1000,_,_,_,_,_). The result of applying the three rules to this query leads to the following result:

r1 ⇒ TRAVEL(_,1000,'Paris',_,_,_,_)
r2 ⇒ TRAVEL(_,1000,'Paris',_,2,_,_)
r3 ⇒ TRAVEL(_,1000,'Paris',_,2,_,X), TRANSPORT(X, 'train')

r1, w1 = 1 : TRAVEL(_,_,departure,_,_,_,_) ← TRAVEL(_,_,'Paris',_,_,_,_)
r2, w2 = 0,7 : TRAVEL(_,_,'Paris',_,nbDays,_,_) ←
TRAVEL(_,_,'Paris',_,2,_,_)
r3, w3 = 0,7 : TRAVEL(_,_,'Paris',_,2,_,_) ← TRAVEL(_,_,'Paris',_,2,_,X),
TRANSPORT(X, 'train')

Figure A5.11: Example of rules for query enrichment

3.3.3 Weighted predicates profile

Another approach for profile representation in the database field is described in [KI05b]. It uses the same formalism for expressing the user preferences as the one used for the profile presented in Figure A5.6 of section 3.1 . A user profile is represented by a set of weighted predicates expressed on attributes of a given schema. The main difference between the approach presented in [KI04a] (see section 3.1) and this of [KI05b] is that in the first one each predicate is assigned a single weight, while in the second one the user interest for a predicate is given using two numeric functions. In the latter case, the user interest for a given predicate *q*, which is expressed on an attribute *A* for a value $u \in D_A$ (domain of the values of A), is denoted by doi(q). It is defined using the functions $d_T, d_F : D_A \rightarrow [-1, 1]$ such as $d_T(u) * d_F(u) \leq 0$:

$doi(q) = (d_T(u), d_F(u)), \forall u \in D_A$.

The sign and the form of the two functions capture the *valence*, the *concern* and the *elasticity* of the user preferences. The valence of a preference is defined by the sign of the values of $d_T(u)$ and $d_F(u)$. A positive value expresses liking, a negative one disliking and a degree equal to 0 indicates indifference. Each function represents one side of the user's concern. The interest on the presence of *u* is expressed by the value of $d_T(u)$, while this of its absence by $d_F(u)$. Finally, elasticity is captured by the form of $d_T(u)$ and $d_F(u)$. For exact preferences $d_T(u)$ and $d_F(u)$ are constant functions while for elastic preferences the values returned by $d_T(u)$ and $d_F(u)$ depends on the degree at which the preference is satisfied. Examples of elastic functions are given in Figure A5.12.

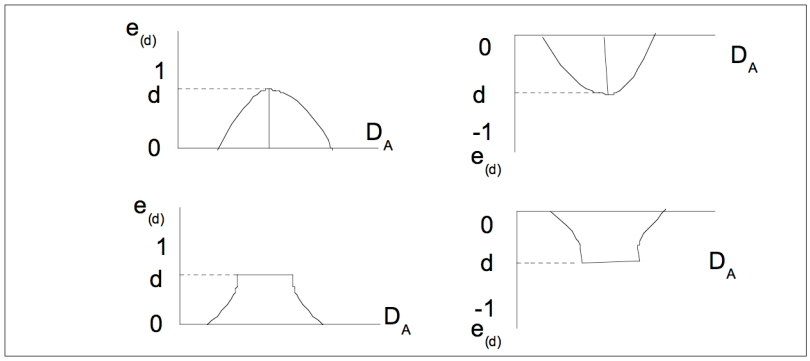


Figure A5.12: Examples of elastic functions

The profile P_e in Figure A5.13, gives possible weights to the predicates of the profile in Figure A5.6 using the functions d_T and d_F. For each selection predicate the user concern of the absence of the value is added (second weight values) and the interest of 2 days travels (predicate e) is given with an elastic function. It returns a value which is at most equal to 0.4.

$\mathbf{P_e}$		
{TRAVEL.idT = TRANSPORT.idT	(1)	(a)
HOTEL.idH = VOYAGE.idH	(0.9)	(b)
TRAVEL.departure = 'Paris'	(0.9, -0.8)	(c)
HOTEL.region = 'centre city'	(0.8, -0.7)	(d)
TRAVEL.nbDays = 2	($e_{0.4}$, 0)	(e)
TRANSPORT.mean = 'train'	(0.3, -0.3)	(f)
TRANSPORT.mean = 'car'	(0.2, 0) }	(g)

Figure A5.13: Extended weighted predicates based user profile

The weighted predicates in the user profile are used for query enrichment. This technique will be detailed in the following sections.

As we mentioned before, most of the preference expression formalisms presented in this section only refer to the domain of interest. The preferences which describe other aspects of the user are usually expressed using attribute-value formalism. Examples of such preferences are latency-recency profiles in [BR02] or the execution cost and result size parameters in [KI05a].

As previous examples have shown, there are as many profile definitions formalisms as application domains and technologies. More than one formalism can be used to express preferences in the same profile. For example most of the techniques presented in this section refers to the domain of interest. But there are formalisms which can be used or adapted for other preference expression than those they are actually describing. For example, the preference composition operators of [Kie02] can also combine predicates based preferences.

3.3.4 Multidimensional user profiles

Since identification and organization of profile knowledge is a key issue to have a global view of data personalization, different attempts have been done to collect and classify this knowledge. For example, P3P [CLM02], as a standard for profile security, has identified three categories of profile knowledge: *demographic attributes* (e.g. identity, age, revenue), *professional attributes* (e.g. employer, job category, expertise) and *behaviour attributes* (e.g. trace of previous queries, time spent at each navigation link).

Another categorization tentative has been done by [AS99] for the digital libraries field. They have identified four categories of knowledge: *personal data* (identity), *collected data* (content, structure and origin of accessed documents), *delivery data* (time and support of delivery), *and behavioural data* (trace of user-system interactions).

Germanakos and Mourlas [GM05] had defined what they named a "New" user profile that is directly related to the multimedia content field. The profile is identified by three categories: *perceptual user requirements* (e.g., visual attention, cognitive process), *traditional user characteristics* (personal, demographic, interests, abilities, needs, previous experiences), and *device characteristics* (displays, memory and storage space capacity, processing power, bandwidth, battery lifetime).

Another profile category which seems to became very important is the cognitive representation of the user. In the Alter Ego SOTA [DEF+05] the problem of personalization and adaptation of communications and applications to individual users is modelled as *USABILITY* concept. The standard ISO 9241 describes usability as a process of three components:

- Effectiveness: *it is the accuracy and completeness with which a particular user can achieve some goal in a particular environment,*
- Efficiency: *it principally refers to the ease of using an application,*
- Satisfaction: *it refers to the comfort and acceptability of the system to its users.*

Consider for example the *efficiency* which refers to all user extended resources, especially the cognitive one. For Loorbach and al. [DEF+05] efficiency can not be achieved without taking into account three intrinsic user characteristics which are physical abilities, cognitive abilities and style preferences.

- *Physical abilities*

It is obvious that efficiency of computer use depend on user abilities and disabilities with regard to the visual perception, the sound perception and the fine movement. For example, physical limitations such as reduced sight and/or hearing, difficulties to distinguish colours (ex. Daltonism), difficulties to make small and precise movements concern much more elderly population segment that represent more than 37 million people in Europe alone. These problems can cause considerable accessibility and ease-of-use limitations. Moreover these limitations may also be caused by the environment and context in which application is used. Such example is accessing information when driving or through a small telephone display. Therefore, it is very important to include such characteristics in the user profile in order to provide more efficient and personalized content to people with physical limitations.

- *Cognitive abilities*

In their State of the art [DEF+05], Loorbach and al. consider cognition as a generic term for all kinds of process that "*go on in our heads*" when we perform everyday tasks. Cognitive process includes attention, perception, memory, learning and language.

- *Attention* is the process of selecting things to focus on, at a given moment. Attention is influenced by signals that human perceive through senses. Adaptive interaction and interface design should present relevant information for particular user goals.
- *Perception* is the process of information acquisition trough senses organs from the environment and the process that transform information into experiences. Vision is the most dominant sense followed by hearing and touch. Knowing user perception specificities can help system/application to present information in adequate and good combined multiple modalities (ex. Text and image).
- *Memory* is a complex process that includes the manner of sorting and recalling information, encoding information into knowledge, the context in which the information is stored, etc. Since people are more able to recognize things than to remember them, and since context in which information is stored influences the ease with which information can be retrieved from memory, it seems very interesting to adapt the information's presentation to the context.
- *Learning* is the process of acquiring "new" competences and knowledge. Adaptive interaction and interface design should present information in ways that take into account the knowledge and skills that the users already have when initially using the system or have acquired through recurrent use.
- *Language*: users may differ in their verbal language competence, either for productive competence (speaking, writing) and receptive competence (listening, reading). Verbal information is more easily understood when the used language is well adapted to the level of information receiver. Hence it is interesting to include user language *preference* and *competence* in his profile.

- *Style preferences*

Obviously people have different preferences for how to process information, how to learn and how to think. This kind of preferences is called cognitive style. Cognitive style is a part of user personality which seems to be decisive when users meet problems in using the information system and in the manner that these problems are solved (finding solution). Including personality data in user profile (such as cognitive style), may compromise the systems. The problem in this case is: *How to collect such personality data that are valid and reliable?* Personality data can neither be extracted from a simple questionnaire frame nor inferred from user behaviour. Thus, such data is measured with specially designed psychological tests. G.E.F.T (Group Embedded Figures Test) and M.B.T.I (Myers-Briggs Type Indicator) are the two widely-used tests measuring cognitive style. It is important to notice that it is unlikely that users accept to do these test (in order to be recognized having a particular style) if they don't see clearly benefits of time and effort they have to invest.

Germanakos and al. [GM05] had been interested in the user's cognitive dimension (called user perceptual requirements) which is included in their "New" User Profile. In order to implement and incorporate visual attentions, cognitive and emotional processing parameters into user profile, Germanakos and al. assume that user perception preferences could be considered as a continuous mental processing, that begin by one stimulus in the visual attention field and go through many cognitive process.

A three-dimensional approach for implementing perceptual preferences is presented in [GM05]. These dimensions are:

- *Learning Style*

In this dimension, it is question to know how users transform information into knowledge. Are they converger, diverger, assimilator, accommodator, analyst, verbalizer or imager? This may allow personalization system to provide the most appropriate learning style.

- *Visual and cognitive processing*

This dimension is composed of *visual attention processing* in which the eye organ plays a key role. Based on the assumption that when performing a cognitive task, while watching a display, the location of users' gaze correspond to the object which is currently analysed, interpreting eye movement seems to be useful to infer, predict and then influence user visual attention. The second component is memory working that refers to the users' abilities on holding information in active state during their interaction with the system. The two final components are *control of processing* which refers to the ability to identify goal-relevant information and *speed of processing.*

- *Emotional processing*

This dimension is composed of parameters that characterise the user's emotional state while interacting with system. These include extroversions, conscientiousness, neuroticism, open to experience, understanding of emotions, regulation of emotions, and self-control.

3.3.5 Ontological User Profiles

By ontological profile, we mean all profiles that are expressed on an ontology or a taxonomy. In *MyNews* [Dem02], users express their preferences over a sample of news taxonomy by rating taxonomy concepts in [-2, +2] scale. The taxonomy is divided into three sections, categories (sport, economic, politic, etc.), locations (America, Africa, Asia, Europe, France, Canada, China, etc.), and people (Bill Gates, Koufi Anan, Nadal, Zidane, etc.). This segmentation enhances the ease of use of such personalized application and make introduction of preferences very pleasant for users. Figure A5.14 shows screenshots corresponding to

categories, locations and people sections. Note that items/concepts are rated using a simple remote control. This prototype was developed by *Accenture technology Labs (USA)*.

Figure A5.14 : Three screen from MyNews User interface

In [SF06], a technique for inferring user preferences using ontologies is presented. Authors argue that an a priori score can be calculated of every ontology-concept based on the location of the concept in the ontology and without using any user information. Then, when a user interacts with a personalized service, he can explicitly rate different interesting concepts, these rating will be propagated all over the ontology in order to update other concept-rates. Another example of ontological profile is given in [SMS07], where both users and products are expressed on the same ontology. Authors had presented a technique that allows performing content-based filtering on such profiles. In this case, user profiles are composed from ontology concepts or branch of concepts, where product profiles are composed only from deeper concepts.

4. Construction of user profiles

This section presents the process of profile construction. The goal of this process is to capture and model the behavioural patterns and profiles of users. This phase is very important in a personalization framework. Indeed, the quality of the entire personalization task depends on the quality of the profile and thus, on the quality of its construction.

In the literature, several papers propose a survey of the profile construction process. Articles by Mobasher and al. [Mob07, Mob05, AM05] present existing approaches for it. In these papers, Web personalization is viewed as an application of data mining and machine learning techniques. The study of these techniques is out of the range of this thesis. An introduction to these techniques from a database point of view can be found in [HK06]. Another survey on *Web personalization* is [EV03]. This work studies lots of commercial tools which are concerned by *Web Usage Mining*, i.e. extracting user profile from usage data from web server log files. In another context, the review paper by Levene and al. is focused on profiling from textual data [PML05]. The profile in this case is a collection of words or expressions. The authors evoke the use of ontologies in profiling and present commercial softwares.

The problem of constructing user profile was addressed in many works in the literature. In fact, almost all the articles presenting an application of personalization tackle the problem of profile construction. Proposed approaches come from different domains such as statistics, information retrieval or data mining. The diversity of existing approaches imposes to point out some characteristics to classify them. For a given personalization application, the choice of the profile building method can be made with respect to several parameters:

- The way the application will provide personalization and the profile model: obviously, the chosen method will have to be able to build a profile respecting the preference language used in the application.

- The real-time aspect of construction phase: building the profiles online implies to use algorithms with bounded execution times.
- The data sources and the way data are collected: on one side, the user can explicitly define its profile or, on the other side, the system can infer it.
- The update of user profiles: when new data are available, profiles can be updated or the entire construction process can have to be executed again.

We can note that these parameters are not independent. For instance, if the profile construction is done online, it is reasonable to think that profile updates have to be incremental: executing the entire process for each update will be too time consuming. The following sections are going to detail these aspects.

4.1 Real-time aspect

The choice here is to decide if the construction of the user profile will take place online or offline. Note that this choice has nothing to do with the use of the profile which is performed online. In the online case, it is necessary to take into account execution time of algorithms. Indeed, when the user interacts with the system, he is not laid out to wait for a long time for each query. Algorithms of this class are called memory based learning algorithms (also know as lazy). In the offline case, the constraint on response times is less crucial. The construction has to be done during the time interval between two successive updates. The choice of algorithm is thus larger in this situation. These kinds of algorithms are called model based (or eager) learning algorithms. It is important to note that most of the works in the literature choose such a construction mode.

As an example of online construction, we can cite [PRGM03]. The authors expose a technique for tackling automatic learning of the user profile which is performed online. This technique summarizes immediately the metadata received by the system and characterizes immediately the user interests. In this approach all possible user profiles are represented by a tree and each sub-tree captures a specific interest.

Another example is presented in [SHY04]. The authors assume that the preferences of each user consist of the following two aspects: persistent preferences and ephemeral preferences. In persistent preferences, the user profile is incrementally developed over time and it is stored for use in later sessions. In ephemeral preferences, the information used to construct each user profile is only gathered during the current session and it is immediately exploited for executing some adaptive process aimed at personalizing the current interaction.

4.2 Implication of the user

This section deals with the way the user is implied in the construction process. At one extremity of the spectrum, the process can be totally manual: the user has to enter his preferences in the language used by the system or using a simplistic interface. At the other extremity, the process can be totally automatic: the system infers user preferences by observing and analyzing the behaviour of the user. Between these extremities, it is possible to identify several intermediate methods. For instance, the user can use a more sophisticated interface to express his preferences which hides the complexity of the underlying preference formalism. As another example, the user can provide explicit feedback as a like/dislike notation of items. Several levels of automation can be used together in the same system. For instance, the system can infer preferences from user behaviour and also give the user the possibility to modify his profile through a more or less advanced interface. The problem is

thus to integrate collected information in the user profile. This difficult problem will be detailed in the next section which deals with data sources.

In the literature, most of user profile construction are doing without any user effort. [Cha00] presents a non-invasive approach to estimating the user's interest in a web page without directly asking the user. But in many others works the user provides feedback to the construction mechanism. An example of such approach is presented in [MWT04] where the user is asked directly to enter or correct his preferences by presenting the values of his profiles.

4.3 Data sources

This section is devoted to the description of the numerous data sources which can be used for the induction of user profiles.

- **Usage data**

The most commonly used data source is without any doubt usage data represented by server log files. These files can be Web server log files in a Web usage mining context or others applications server log files. They contain the trace of the user interactions with the system which represents the fine-grained behaviour of users. These data can have to be transformed and aggregated in order to be used efficiently in the construction process. For example, time spent on each page can be calculated and the longer a user spent on a page, the likelier the user is interested in the page. If a page is not interesting, a user usually jumps to another page quickly.

Mobasher has largely explained this type of data source in his work [Mob05] and many others works in the literature have used this type of information in there mechanisms of user profile construction such as [Cha00, MWT04]. For example, in [MWT04] the authors present an algorithm to build profiles denoted as *Non Obvious Profile* inferred by the user's behaviour during his visits. The relevant data to be used in the algorithm is extracted from usage data such as various log files generated by the web server. These data are retrieved and stored in database for further processing.

- **Content data**

Content data represents another kind of data sources. They encompass items and documents to which the user has access. Content data also include available metadata about items and documents. The data sources used to deliver or generate this data include static HTML/XML documents, images, video clips, sound files, dynamically generated page segments from scripts or other applications, and collections of records from operational databases. The site content data also includes semantic or structural meta-data embedded within the site or individual pages, such as descriptive keywords, document attributes, and semantic tags. Finally, the underlying domain ontology is also considered part of the content data.

In [MC07], a method to create user profiles from tagging data is exposed. The authors focus on the tags which were employed by a certain user. They treat them as a continuous stream of information about a user's interests in order to creating a user profile.

- **Structure data**

Structure data which describes the organization of items and documents can also be used during the profile building process. As an example of such kind of data, we can consider

the site map of a Web site. It shows the organization of Web pages and expresses how the site designer wants to present information to the user.

In [Lie95], the author thinks that if the page is interesting to the user, he is likely to visit the links referenced by the page. Hence, a higher percentage of links visited from a page indicates a stronger user interest in that page.

- **User data**

User data are a valuable data source. It encompasses demographic and identifying information about users, user ratings on various items or past purchases. Note that capturing automatically such data can be difficult and thus the user is generally implied in their collect (by filling a form for instance).

An example of user data is the bookmarks which can serve as a quick access point for interesting URL's chosen by the user. With a few mouse clicks, a user can easily jump to URL's in his/her bookmarks. Another example is the history of the user's queries in the current session and in the past. The history of the current session allows the user to go back and forth between the pages he/she has visited. In addition, a global history maintains the timestamp of the last time each page is visited. A higher frequency and more recent visits of an URL indicate stronger user interest of that URL.

An example of system which uses this user data is presented in [SHY04]. A profile construction algorithm is developed which uses as input the user's browsing history of web page from N days ago.

- **Using multiple data sources**

Obviously, it is tempting to use several data sources to build a more accurate user profile. This natural idea leads to lots of difficulties. Indeed, this problem is related to the problem of data integration in the database field. An introduction to data integration, mediation and data warehousing, can be found in [HRO06, Len02]. The need to transform all these knowledge into the same formalism (the one chosen for expressing preferences in the user profile) is an example of such a difficulty.

Data warehouse based integration approaches were proposed in [ZXH98, BM98]. In [ZXH98], the server log data are stored in a multi-dimensional data structure for OLAP analysis. This allows the analysis to be performed, for example, on portions of the log related to a specific time period or, at a higher level of abstraction, according to the site structure. In [BM98], different information is stored in an integrated *web log data cube* which incorporates customer data, product data and domain knowledge.

The system presented in [AY02] does not really integrate several sources. Indeed, it uses the knowledge from different sources at different stages of the personalization process. Content data are acquired from documents using text mining techniques and site-specific categorization. Usage data comes from log files and allows building user profile using clustering algorithms. Usage data are also used to compute statistics for access frequencies to documents.

In the SEWeP system, the integration is performed by enriching classical web log files with content data [EVV03]. The obtained conceptual logs are then used as input for web usage mining. Firstly, keywords are extracted from web pages using a combination of IR techniques based on term frequencies and web links. Keywords are then mapped to the concepts of a domain-specific taxonomy using a thesaurus. Finally, each records of the web server log which was beforehand pre-processed (session identification, ...) is enriched with the discovered categories.

In [JZM04], the integration of usage and content data is based on Probabilistic Latent Semantic Analysis [Hof99]. This approach uses two matrices: the former expresses the existence of a pageview (web object) in a session (user) and the latter describes the relationship between attributes (semantic knowledge) and pageviews. From these matrices and using an iterative algorithm called Expectation-Minimization [DLR77], some probabilities are estimated and used to compute a total likelihood which integrates knowledge from both data sources. An extension to this work presenting an alternative probabilistic model is proposed in [JZM05]. Mobasher et al. also present a framework to integrate semantic knowledge (content description with ontologies, structure ...) with web usage mining [DM05].

4.4 Construction methods and algorithms:

In this section, we are going to present methods and algorithms used in the literature for the construction of user profiles which will be grouped in four approaches.

4.4.1 Machine learning approaches

Machine learning is concerned with the design and development of algorithms and techniques that allow computers to "learn". Many classification algorithms exist which are divided into supervised and unsupervised. In supervised classification we have a set of training examples which are known to belong to one of a given set of classes and the aim is to learn the characteristics which place particular items in one of the classes. By contrast, in unsupervised learning, no predetermined classes exist and the aim is to find discerning characteristics which can divide a single relatively heterogeneous group into a number of relatively homogenous groups. The literature offers many machine learning algorithms and methods to build a user profile which will be presented in the following paragraphs.

Decision tree is a supervised classification technique. It is a predictive model that is a mapping from observations about an item to conclusions about its target value. More descriptive names for such tree models are classification tree (discrete outcome) or regression tree (continuous outcome). In these tree structures, each node effectively asks a question about the item in question and each branch from the node represents an answer to the node's question. The author in [Cha00] thinks that it's possible to identify patterns in pages that constitute the user's interest. For example, some words or phases are of interest to the user. Given a set of labelled (interesting or not interesting) pages, the author apply learning C4.5 algorithm to induce classifiers that predict if a page is of interest to the user. These classifiers are called *Page Interest Estimators*. In C4.5, a decision tree is built by repeatedly splitting the set of given examples into smaller sets based on the values of the selected attribute. An attribute is selected based on its ability to maximize an expected information gain ratio. To find out if a user is interested in a page, one way is to ask the user directly (invasive approach), another way is to monitor the user's behaviour and evaluate the user's interest (non-invasive approach) using all types of data sources such as user data and usage data.

Association rule learners are another strongly used method to build a user profile. They are used to discover elements that co-occur frequently within a data set consisting of multiple independent selections of elements and to discover rules, such as implication or correlation, which relate co-occurring elements. As with most data mining techniques, the task is to reduce a potentially huge amount of information to a small understandable set of statistically supported statements. The SEWeP system [EVV03] generates user profiles as a set of association rules. The particularity of this system is based on the fact that rules include URI and categories extracted from enriched logs. This allows to exhibit rules between different levels of abstraction (URI or taxonomy).

Another machine learning method often used is the Genetic algorithms (GAs) which are iterative search techniques based on the spirit of natural evolution. By emulating biological selection and reproduction, GAs can efficiently search through the solution space of complex problems. Basically, a GA operates on a population of candidate solutions called chromosomes. A chromosome, which is composed of numerous genes, represents an encoding of the problem and associates it with a fitness value evaluated by the fitness function. This fitness value determines the goodness and the survival ability of the chromosome. Hence, GAs are powerful tools for optimizing the process of classification and construction, in particular when the domain knowledge is costly to exploit or unavailable.

In BEAGLE [Fer95], the author builds a population of user profiles which represent the best subset of keywords for distinguishing relevant documents from non-relevant ones. Other work using GAS is presented in [MVL99] where a user profile is built from the user preferences, represented by a population of chromosomes. Each chromosome is a vector of fuzzy genes and each gene is represented by a fuzzy number of term occurrences that characterizes preferred documents. Hence, the user profile is built from the population of the GA, leaded by the best chromosome.

Support vector machines (SVM) are a set of related supervised learning methods used for classification and regression. They belong to a family of generalized linear classifiers. SVM is a binary classifier algorithm which represents documents using the vector space model and treats each document as a point in a multi-dimensional feature space. The task is then to find a hyper-plane which separates all the points belonging to one class from all those belonging to another. A special property of SVMs is that they simultaneously minimize the empirical classification error and maximize the geometric margin. Hence, they are also known as maximum margin classifiers.

In [SCZ05], the authors use this method to construct the user profile based on the profile ontology. They develop a system where a user only has to provide a directory in which there are some documents he is interested in. These documents can be the resources downloaded recently from the internet or the materials written by him. They provide profile ontology organized as a hierarchy tree with its leaves as the elementary categories and the documents given in the user directory are classified into those elementary categories. SVM is used as the classification method to map the document feature space to the user profile space because it has overwhelming effect in learning from small training sets. Hence, SVM is used to classify the documents in the user given document collection into the elementary categories. The number of documents in each category is labelled as each leave's weight in the tree data structure. The other nodes' weights are obtained in the tree by making each node's weight equal to the sum of its sub nodes' weight. Thus all the nodes in the tree have their weights for describing their relative importance in the user profile. Finally, the user profile vector is formed by choosing the N biggest nodes to be normalized.

In [JZM04], the approach is based on a probabilistic model. The probability estimates generated during integration of usage and content data are used to characterize web user segments. These probabilities allow first to select sessions which are similar according to user behaviours. These sessions are then aggregated to build a vector of pageviews which describes the behaviour of the user segment. The probabilities allow computing the weights of each pageview in the vector. Others examples which use a probability model are also mentioned.

4.4.2 Graph theory

Graph theory is the study of graphs, mathematical structures used to model pair wise relations between objects from a certain collection. A graph refers to a collection of vertices and a collection of edges that connect pairs of vertices. A graph may be undirected, meaning that there is no distinction between the two vertices associated with each edge, or its edges may be directed from one vertex to another.

In [Cha00], the authors present *Web Access Graph* which is a weighted directed graph that represents a user's access behaviour. Each vertex in the graph represents a web page and stores the access frequency of that page. If the user visited page A followed by page B, a directed edge exists from page A to page B. The intensity of a vertex indicates the interest level of the web page.

Another example of works that use the graph theory is presented in [MC07]. The authors think that if two tags are used in combination by a certain user for annotating a certain bookmark, there is some kind of semantic relationship between them. The more often two tags are used in combination, the more intense this relationship is. They represent a graph with labelled nodes and undirected weighted edges in which nodes correspond to tags and edges correspond to the relationship between tags. Each time a new tag is used, a new node for this tag is added to the graph. Each time a new combination of tags is used, a new edge with a weight of one between the corresponding nodes is created in the graph. If two tags co-occur again, the weight for the corresponding edge is increased by one. The graph is created by parsing the tags for all items in the bookmark collection and applying the technique described above. A user profile is derived from the resulting graph by selecting the top k edges with the highest weights and their incident nodes. The problem of the approach presented by [MC07] is that the age of bookmarks and their temporal ordering is not considered. This issue is addressed by *Add-A-Tag algorithm* which is also presented by [MC07]. This algorithm extends the last approach with the *evaporation technique* known from *Ant colony optimization algorithm (ACO)* [DDS02]. ACO is a probabilistic technique for solving computational problems which can be reduced to finding good paths through graphs. Evaporation is a simple method to add time-based information to the weights of edges in a graph. Each time the profile graph is updated with tags from a newly added bookmark, the weight of each edge in the graph is decreased slightly by removing a small percentage of its current value. When creating the profile graph for the adaptive approach by parsing the tags for all items in the bookmark collection, it is necessary to start parsing from the oldest item and to process the items in the same temporal order as they were added to the bookmark collection. Again, the user profile is created by extracting the top k edges with the highest weights and their incident nodes from the profile graph.

4.4.3 Weighted terms

Weighted terms is another important approach used in order to build the user profile, it enclose different techniques such as Boolean weighting or frequency weighting. The central idea governing all these techniques weighting is that the occurrence of terms within documents characterise the user profiling to which the document belongs. Hence, the more frequently a term occurs in a class, the more indicative it is of that class. In [MC07], the authors present a naïve approach that aggregates the data in such a way that the interests of the user are reflected according to their intensity. The more often a certain tag is used, the higher the interest of the user in the corresponding topic. Therefore, the simplest method for creating aggregated data for a user's bookmark collection is to count the occurrence of tags. The result of this computation is a list of tags which is ranked according to tag popularity.

4.4.4 Fuzzy approaches

Fuzzy measures can be introduced for two different uses: either they can represent a concept imprecisely known (although well defined) or a concept which is vaguely perceived such as in the case of a linguistic variable. Usually, most of the works in the literature consider the fuzzy as a complementary technique for the others techniques largely described above. Hence, other method can be extended by the introduction of fuzzy concept in order to obtain a more accuracy user profile.

In [MVL00], the authors use a fuzzy classification with a genetic term selection process which can provide a better utilization of valuable knowledge for genetic algorithms in order to get an improvement of the quality of the current and near future information needs in the areas of interest to the user. A gene in the chromosome of the genetic algorithm is defined by a term and a fuzzy number of occurrences of the term in documents belonging to the class of documents that satisfy the user's information need. In this way, the terms that allow the system to discern between good and bad documents are selected and stored as a part of the user's profile to be used in future queries to the system. The fuzzy classifier in this work [MVL00] implements an inductive derivation of the current, experience based, interest profile in terms of an importance weighted conjunction of genes.

In [PRGM03], the authors affirm that it's possible to generate a tree-based organization of fuzzy concepts based on the SAINTETIQ [RM02] model. The learning task is performed on-line, summarizing the metadata received by the system and characterizing the user interests in this tree as several sub-trees of this summary hierarchy.

4.5 Evolution and updates

An important aspect which is directly related to the construction of the profile is its maintenance. Indeed, the simplest approach to insure profile update is to execute again the construction algorithm from updated data sources. At the opposite, a modification of a data source can lead immediately to an update in the user profile. This situation can happen for instance in a system using an online construction process. In this section, we are going to present methods and algorithms used in the literature for the update and revising of user profiles.

Collaborative techniques are often used in this context. The main idea is that the best way to complete and to maintain the accuracy of a user profile is to use the information provided by the similar users. The authors of [SHY04] consider a user-term weights matrix. They think that it's possible to apply collaborative filtering algorithms to predict a term weight in each user profile. In other words, since each user profile is computed based on term weights in a web page the user has browsed and the browsed pages are different according to each user, the profile is constructed in the form of a user-term weights matrix with missing values. It's clear that a more accurate user profile is constructed since these missing values are predicted using the algorithm in collaborative filtering which is explained in the following steps:

1- Weight all users with respect to similarity to the active user. This similarity between users is measured as the Pearson correlation coefficient between their term weight vectors.

2- Select n users that have the highest similarity to the active user. These users form the neighbourhood. A weighted aggregate of their term weights is used to generate predictions for the active user in the coming step 3.

3- Compute a prediction from a weighted combination of the neighbour's term weights. Predictions are computed as the weighted average of deviations from the neighbour's mean.

We can note also another approach to collaborative techniques based on graph-theoretic denoted as *Horting* [Grc04]. It involves building a directed graph in which vertices represent users and edges denote the degree of similarity between them. If we are trying to predict user *u* rating of item *i*, we need to find a directed path from user *u* to a user who has rated item *i*. By using linear transformations assigned to edges along the path, we can predict user *u* rating of item *i*. No other user along this path rated item *i*. This means that *Horting* can explore transitive relationships between users.

The second technique which is also largely used is the feedback mechanism. The user is asked directly to enter his preferences by presenting the values of the profile and asking the user to verify or correct them. The authors of [MWT04] build up another profile called the feedback profile from the feedback given by the user. The initial profile is updated with the latest session information together with the start of the feedback mechanism. There are various ways in which the feedback mechanism can be started, for example it can be triggered on certain events or can be activated after a change is detected.

Genetic Algorithm is a technique used to address the problem of inaccurate user's profiles [CS01]. Based on the Genetic Algorithm (largely explained in the previous point), the authors use the users' relevance feedback to improve the profiles automatically and to provide the feedback only at the initial step of the evolution. Many systems such as Adaptive Soft Query System [CS01] uses a GA based learning mechanism to improve the user profiles through users' feedback. This learning mechanism is an automatic and off-line process. It employs GA for improving the user profile by decoding the best chromosome to replace existing user profiles in the database after its evolution. User involvement is only needed for providing the relevance feedback as the goal of GA prior to the beginning of evolution. The authors note that the learning mechanism is only triggered by the user feedback and not needed during querying processing. In this mechanism, the chromosomes represent possible user profiles of a specific user and each gene corresponds to a record in the user profiles. Two types of records are involved in the genes. One is user confidence information and the other is a user fuzzy cut value. This method can guarantee a one-to-one mapping of profiles to chromosomes. That is, a chromosome will be decoded to one and only one legal user profile and a user profile will be encoded to one and only one chromosome. Once a user offers his feedback to trigger the learning process, the learning mechanism first encodes the corresponding user profile to a chromosome and randomly generates other chromosomes as the initial population. Subsequently, GA iteratively discover better user profiles until it achieves the terminal condition such as the fitness value of one chromosome being one or more. In the end, the learning mechanism decodes the best chromosome to a user profile for replacing the current user profile in the database.

5. User profile exploitation

In this section, we present the manner that user profiles and preferences are used in order to provide personalized services and content. We assume that user profiles are built, and content sources are available. We observe four scenarios of profile usage: introducing preferences into query languages, query enrichment, query rewriting and matching the user profile with other profiles or with content.

5.1 Introducing preferences into the query languages

In order to take into account user preferences, some query languages have been extended with concepts which are very useful for integrating the user profile content into the query. They have been motivated by research on fuzzy data bases [BP95, LNR+01, Roc03] and flexible query execution [SP04]. Thus, an interesting extension of SQL, called

PRFERENCE SQL [Kie02], enables expressing an important number of preferences and consequently is well adapted to data personalization. To illustrate the concepts of the languages which enable personalization we take three cases: introduction of context dependant operators, introduction of preferences into the selection criteria and introduction of partial orders between preferences.

Introduction of context dependant operators

In some cases, it is often interesting to express approximate criteria such as "around 80 m^2, not very expensive, well situated, close to Versailles". This kind of queries is well adapted to the usage of the profile content in the sense that the semantics of each term depends on the user. If the user profile enables defining those terms' semantics, by fixing the distance or the interval size for example, one can have a good flexibility in data access. In this case, if two users which have different profiles, submit the same query, they will be given different results. Some systems such as PREFERENCE SQL [Kie02] propose approximate operators such as ARROUND and BETWEEN, but their definition is fixed. Thus, the same query will give the same results independently of the user who issued it.

Introduction of preferences into the selection criteria

All query results do not have the same degree of interest to the user. One of the techniques which can be used to retrieve the best results is to introduce a preference on the attribute values. In [Cho02] preferences are expressed using preference formulas (see Figure **A5.8** for an example). These formulas are then used by the WINNOW operator to find the best tuples. In other words, two tuples appear in the result if they have the same values or if they are indifferent (none of the tuples is best compared to the other).

Another example of preference operator is the SKYLINE [BKS01]. To use it, the SQL syntax is enriched with an additional clause of the form:

`SKYLINE OF [DISTINCT]` d_1 `[MIN | MAX | DIFF], ...,` d_m `[MIN | MAX | DIFF]` where d_i are skyline dimensions.

Each skyline dimension is an attribute of one of the relations in the queried schema. The tuples returned by the SKYLINE are these not dominated by other tuples. A tuple t_1 dominates another tuple t_2 if t_1 is as good as t_2 in all dimensions and better in at least one dimension.

Another approach to expressing preferences among selection criteria is used in PREFERENCE SQL [Kie02, Kie05]. It enables defining positive (POS) or negative (NEG) sets of values for a given attribute and if these sets are not sufficient, explicit (EXPLICIT) preferences among a couple of values can be expressed. For more details about the definition of POS/NEG/EXPLICIT preferences see section 3.3.2.

Introduction of partial orders between preferences

Some query languages enable defining preference hierarchies which specify partial orders between selection criteria. In PREFERENCE SQL [Kie02, Kie05] for example, this partial order is introduced with the help of two clauses: PREFERING and CASCADING. The first clause (PREFERING) fixes the initial choice of selection criteria. That is, criteria which have to be considered immediately after the query selections. The second clause (CASCADING) introduces criteria which are less important than the previous ones. Consider for example the view TRAVEL in Figure A5.8. If the a user wants to go *in Barcelona*, prefers travels which cost less than 1500 euros and if many travels are available decides on the number of days, his need can be expressed with the following query:

```
SELECT *
FROM TRAVEL
WHERE arrival = 'Barcelona'
      PREFERING price < 1500
           CASCADING HIGHEST(nbDays)
```

Another approach which enables defining hierarchical selection criteria is presented in [LL87]. In this approach, selections which have the same importance are introduced using the PREFER clause while for less important ones the clause FROM WHICH PREFER is used. In this case, the selection criteria introduced by the preference clauses can be seen as filters which are applied to the results of the query without these criteria. When applying preference criteria, the order established by the hierarchy is respected. Each hierarchy level contains selections which have the same importance. When applying criteria from a given level, only results which satisfy the highest number of criteria are kept for further filtering. If there are not any results when applying the selections from a given level, the operation is invalidated and the next levels are considered.

Even if the concept of user profile does not exist in the approaches which extend the query languages, these languages are a step toward data personalization as some domain of interest data can be directly expressed on them.

5.2 Query Enrichment

Query enrichment consists in integrating elements of the user profile into the user's query. This technique, commonly used in information retrieval systems [FS01, LBE+98], is very recent in the database domain.

An interesting method defined recently is the one of Koutrika and Ioannidis [KI04a, KI05b]. To some extent, this method appears as very close to the view mechanism. Indeed, the user profile is defined as a list of disjunctive predicates, including selections and joins. Predicates are associated with weights, ranging within [0,1] interval and representing their relative importance with respect to the user preferences. Among this ordered set of predicates, the user can also specify another type of preference:

- K: the number of predicates he wants to be taken into account in the enrichment process,
- M: the number of mandatory predicates among the K ones and
- L: the minimum number of non mandatory predicates among the (K-M) remaining ones that each result have to satisfy.

Given such a profile, the query enrichment process consists in reformulating the initial user query by adding predicates from this profile. It consists in two phases: (i) predicates selection and (ii) predicates integration to the query.

Predicates selection

The first step of query enrichment consists in selecting the « *top k* » profile predicates which will be used to enrich the user query. In order to be selected, each predicate has to be related to the user query and not conflicting with it. A profile predicate is related to the user query if one of the following conditions holds: (i) the predicate is expressed on an attribute which belongs to one of the query's relations or (ii) the predicate can be related to the query using join predicates from the user profile. A predicate is conflicting with a given query if it is conflicting with at least one query predicate. Example of conflicting predicates are "price > 10" and "price < 5" because they can not be satisfied together.

Once the profile predicates which satisfy the two conditions have been identified, the Top K ones are selected according to their degree of interest. The degree of interest of a given predicate is a function of its weight, the number of join predicates necessary to relate it to the query and the weights of these joins. It is assumed that the weight of a predicate decreases when its distance to the query (in term of join predicates) increases.

Predicates integration to the query

The second step of query enrichment consists in integrating the Top K profile predicates to the query. Two strategies can be followed to do this: generation of a single query or generation of multiple queries.

The first strategy consists in four steps: (i) make all mandatory predicates as one conjunctive clause, (ii) make a conjunctive clause with each possibilities of L among (K-M) remaining predicates, (iii) make a disjunctive clause of all these possibilities, and finally (iv) add these new clauses as conjunctives subqueries to the initial user query.

The second strategy is to formulate a set of K-M partial queries where each partial query is obtained by adding to the initial query the conjunction of the mandatory predicates and one non mandatory predicate. The results are computed by making the union of all partial queries, grouping by all attributes of the SELECT clause and excluding groups which contain less than L rows.

More detailed description of the query enrichment can be found in chapter 4 of the thesis.

5.3 Query rewriting

The query rewriting process consists in transforming the user query expressed on the virtual schema so that it can be evaluated on the data sources. It aims to determine contributive data sources for query execution and to use their definitions to reformulate the query. Surveys on query rewriting can be found in [Hal01, CLL01].

In the literature, personalized query rewriting is viewed as a problem of source selection with respect to their quality [NFS98] or as a problem of traversing paths in the graph which represents the data sources [VRM+06].

In [NFS98] sources are selected according to their quality. The quality of a given source, called efficiency in the paper, depends on three quality criteria: understandability, extend and availability. The first criterion (understandability) measures how well a source presents its information to users. It is expressed on a 1 (bad understandability) to 10 (excellent understandability) scale. The second criterion (extend) is the average length of the single piece of information in a given source. Examples of extend metrics are the average number of fields and the average number of words over the documents. The third criterion (availability) is the probability that a feasible query is answered in a given time range. Each source is assigned a quality score for each of the quality criteria. Thus, the efficiency of a source is a weighted sum of its quality scores. Computing the efficiency of a given source is defined as a linear program which attempts to maximize the weights of the criteria where the source scores well. All sources which have the best quality score for at least one quality factor have an efficiency equal to 1 (the maximal value). Indeed, assigning a high weight to the quality criterion where the source has the best value and a weight equal to 0 to all other criteria is a weighting where the source has the best efficiency.

In [VRM+06] query rewriting is seen as a graph traversal problem. This approach is defined in the semantic Web context where the data model is comprised by three levels: ontology, physical and data. Each of these levels is modelled using a graph. The ontology

graph represents concepts and associations which model a given domain. The physical graph models data sources and links among them. The data graph contains entries published by data sources and references among them. Relationships between these graphs are expressed with 4 types of mappings:

- mappings which define each ontology concept in terms of the data sources,
- mappings between the associations in the ontology level and the sources' links,
- mappings between the data sources and the entries published by the data sources,
- mappings which express the links between sources in term of references between entries.

In this approach, a query is expressed on the ontology level using regular expressions. The main problem discussed in [VRM+06] is how to find the Top K query rewriting with respect to a given metric M. A query rewriting is a path in the source graph which satisfies the regular expression of the initial query. In order to find the Top K paths, each source path is characterized according to a given metric. The authors present three metrics: target object cardinality, user preference and cost of evaluation a path. The target object cardinality of a given path is defined as the number of entries in the final source which can be accessed from the initial source. The user preference for a path is equal to the average of the user preferences for all objects of the path, assuming that the user have given a preference weight in the range [0, 1] for each node and edge in the source graph. Finally, the cost of evaluation a path is the sum of the costs of evaluating each link in the path. The cost of evaluating a link is computed in terms of the number of intermediate entries generated by the link during the execution of the query.

5.4 User-User and User-Content Matching

In many approaches the user profile exploitation consist in calculating its distance/similarity according to the items or to the profiles of other users. Both user to user and user to content matching need a distance or similarity function. This section present some commonly used distance and similarity techniques before describing the user to user and the user to content matching.

Measures of similarity and distance

We find in the literature many similarity and distance measures. These measures differ in the type of concepts they compare: vectors, graph etc.

A commonly used approach which emerge from the Information Retrieval (IR) field is the Vector Space Approach where documents and user queries are represented by weighted-term vectors. The similarity between a document and a query is measured as a correlation between their vectors. The most used similarity functions are the Pearson correlation [BHK98] and the cosine measure [Kna05].

Another approach where similarity functions are widely used is the graph matching. There exist many techniques to calculate the similarity between two graphs. Bengoetxea [Ben02] propose a classification of all graph matching techniques into two main classes, exact graph matching and inexact graph matching. Exact graph matching consists in identifying an *isomorphism* (bijection) between nodes of two graphs. When there is not such isomorphism, an inexact graphs matching is computed. There are many measures which quantify the distance between two graphs. Examples of such measures are the number of modifications

necessary to transform on graph to the other [Ben02, Mes95] and the maximum common sub-graph [FV01, Bun97].

When graphs contain concepts, it is possible to evaluate the semantic distance between two concepts. This distance is usually measured as the length of the shortest path between concepts [Kna05, Ben02].

User Profiles Matching

Profile to profile matching is used in two cases: (i) to create communities that share some characteristics, preferences, needs, and interests, or (ii) to predict some user preference based on the behaviour of other users. These two possibilities are known respectively as segmentation and collaborative filtering.

Segmentation consists in dividing users in categories by minimizing the similarity between users of two groups and maximising the similarity between users in the same group. According to [DEF+05], segmentation can be done according to different categories of user characteristics including geographic position, demographic data, behavioural data and cognitive descriptions.

Collaborative filtering [BHK98, Paz99, PML05, and Pot04] is one of most used techniques in recommendation systems. It's based on the assumption that users with similar behaviour may have similar interests. The basic idea is to infer missing information in one profile based on other profiles that have been detected to be similar. Similarity between the users is evaluated based on their ratings on items or their behavioural data.

5.4.1 Matching User Profile to Content

In order to compare a user profile and an item, they have to be represented with the some formalism. For example in *MyNews* [Dem02] both user preference and content metadata are defined over the same taxonomy and in [SMS07] and [MSR04] an ontology is used to reconcile the user and the items representation.

In their ontological-based content-based filtering, [SMS07] propose a solution to the problem of ontologies matching. An item's profile consists of a set of ontology concepts which represent its content. The concepts representing an item are the most specific ones in a certain branch of the hierarchy. For example, if an item deals with 'sport' and specifically with 'football', it is represented with 'football' only. Obviously, an item may be represented with several concepts according to its content; same assumption is done in MyNews project [Dem02]. For example, an item's profile may include the concepts *politics*, *football, basketball,* and *rebellions* which may belong to different ontology level. Finally, item concepts are not assumed weighted. A user content-based profile consists of a weighted list of ontology concepts representing his interests [SMS07, Dem02]. Contrary to item's profile, the user profile may consist of many concepts, each appearing in different levels of one branch. For example, a user's profile may include 'sport' only, or 'sport' and 'football' (where 'football' is a child concept of 'sport'). The similarity between the user profile and an item's profile is based on the proximity principle stating that the distance of two ontology concept is directly related to their similarity. In fact, similarity between two concepts c_1 and c_2 is higher if they are closer [Kna05].

6. Conclusion

Although personalization itself is still arguably in its embryonic stage, it can be seen that techniques that are already applied are highly varied due to their foundations in more mature domains. In this annexe, we have presented a comprehensive view of the entire

personalization process. After having exposed motivations and illustrative examples, we have detailed the keystones of this domain : preference and profile models, profile construction and profile utilisation. All these sections have been organized to point out the main characteristics of each aspect of personalization.

While researches into personalization has led to a number of effective algorithms and commercial success stories, a number of challenges and open questions still remain.

A key part of a personalization system is the user model. Commonly used user model are still rather simplistic (set of keywords, etc.). More expressive models need to be explored. This kind of model would probably be multidimensional to encompass all aspects of the user. The formalism to express preferences is still to define.

In particular, personalization system commonly used today lack in their ability to model user contexts and dynamic of user profiles. Indeed, the environment of the user when he interacts with the system has an influence on the expected results. In a similar way, information characterizing the user are dynamic and evoluate. A formalism to express user context has to be proposed and its utilization has to be explained. Identifying changes in the user interests and needs and adapting to them is a key goal of personalization.

Another important aspect is the support for various applications with the same framework. Actually, for each application, a specific framework is proposed. Some efforts have been done to point out more generic architecture in the web usage mining context for instance. However, no proposed framework covers the personalization needs of the majority of applications for a given domain. One difficult task would be to conciliate applications with different requirements (real time personalization, etc.).

A general improvement of personalization systems would be to take into account more semantics in the process. This aspect has to be considered in the construction of the profile (and thus in the user model) and in the use of the profile. One way to deal with this task is perhaps to consider some results coming from the domain of data integration in the database field. Indeed, some problems related to semantic were already studied in this context.

Finally, the ultimate goal of personalization being the satisfaction of the user, some evaluation metrics have to be developed. Current evaluations of systems are based on the accuracy of predictions and the relationship between accuracy and user satisfaction is not obvious. In addition, more business oriented metrics need to be defined to measure the economic benefit of personalization systems.

The contributions of this thesis are in user modelling, user profile exploitation and personalisation algorithms evaluation.