Règles d'Association Triadiques pour la recommandation et l'enrichissement de requêtes décisionnelles

Sid Ali Selmane*, Fadila Bentayeb*, Omar Boussaid*

*Laboratoire Eric Université Lyon 2, France
Prénom.Nom@univ-lyon2.fr

Résumé. Cet article décrit un nouveau processus de personnalisation de requêtes décisionnelles à travers une nouvelle approche d'extraction de règles d'association triadiques. Ce processus exploite les fichiers log des utilisateurs et comporte cinq étapes : (1) génération d'un contexte triadique à partir des fichiers *log* de requêtes d'un serveur d'analyse *OLAP* ; (2) passage d'un contexte triadique vers un contexte dyadique ; (3) production de règles d'association dyadiques conventionnelles ; (4) génération d'un ensemble de règles d'association triadiques par *factorisation* des règles dyadiques. L'avantage de ces règles est qu'elles sont moins nombreuses et plus compactes que les règles dyadiques et qu'elles véhiculent une sémantique plus riche. Et enfin, (5) exploitation de ces règles triadiques pour la personnalisation des analyses OLAP. Nous avons développé un prototype logiciel *P-TRIAR* (*OLAP Personalization based on TRIadic Association Rules*) permettant d'extraire deux types de règles à partir de fichiers log de requêtes. Le premier type de règles servira à la recommandation de requêtes par la prise en compte de l'aspect collaboratif des utilisateurs lors de leurs interrogations de l'entrepôt de données alors que le deuxième type de règles permettra l'enrichissement des requêtes utilisateurs.

1 Introduction

Les utilisateurs des systèmes *OLAP* (*On-Line Analytical Processing*) formulent des requêtes décisionnelles pour répondre à des besoins d'analyse spécifiques pour l'aide à la décision. Les outils OLAP sont connus pour être intuitifs car leurs utilisateurs finaux ne sont pas forcément informaticiens. Cependant, la grande volumétrie des données et la complexité des requêtes d'analyse qui impliquent beaucoup d'agrégations rendent plus difficile la tâche d'analyse aux utilisateurs. Il est donc nécessaire d'offrir à ces derniers des solutions mieux adaptées à leur mode de raisonnement à travers des processus de recommandation et d'enrichissement de leurs requêtes d'analyse. Ce processus est appelé *personnalisation*. Dans cet article nous proposons un nouveau processus de personnalisation de requêtes d'analyse dans un entrepôt de données. Nous nous intéressons particulièrement à la recommandation collaborative et à l'enrichissement de requêtes décisionnelles basé sur les fichiers log des utilisateurs.

Les travaux de personnalisation qui exploitent les fichiers log de requêtes utilisent dans la majorité des cas les *itemsets* fréquents (Khemiri et Bentayeb, 2013) et les règles d'association

(Veloso et al., 2008). Cependant, le très grand nombre d'*itemsets* fréquents et de règles d'association obtenus rendent la tâche de personnalisation plus ardue. Contrairement à ces approches, le travail que nous proposons se base sur un autre type de règles, plus compactes, appelées règles d'association triadiques. Ces règles ont une sémantique plus riche que les règles d'association classiques car elles sont formées en plus de la prémisse et de la conclusion, d'une condition qui s'ajoute à la règle. Notre processus de personnalisation est composé de cinq étapes :

1. Modélisation des fichiers log du serveur d'analyse *OLAP* avec un contexte triadique. Il sera formé de trois ensembles : l'ensemble des utilisateurs, l'ensemble des requêtes, l'ensemble des attributs issus de la clause SELECT (descripteurs et mesures) des requêtes et d'une relation ternaire qui lie ces trois ensembles.
2. Passage d'un contexte triadique (tridimensionnel) vers un contexte dyadique (bidimensionnel). Cette transformation se fera par l'aplatissement (projection) de l'ensemble des utilisateurs sur l'ensemble des attributs des requêtes.
3. Production de règles d'association (RA) dyadiques conventionnelles de la forme *(prémisse* $\rightarrow conclusion$).
4. Génération de deux types de RA triadiques par *factorisation* des règles dyadiques de la forme *(prémisse*$\rightarrow conclusion)_{(condition)}$.(cf. Section 4.1)
5. Exploitation des RA triadiques obtenues pour la personnalisation d'analyses OLAP, soit pour la recommandation, soit pour l'enrichissement de requêtes.

Pour valider notre approche, nous avons développé un prototype logiciel *P-TRIAR* (*Personnalization based on TRIadic Association Rules*) permettant d'extraire deux types de règles à partir de fichiers log de requêtes. Le premier type de règles aura pour vocation la recommandation de requêtes par la prise en compte de l'aspect collaboratif de l'utilisateur lors du processus d'interrogation. Cette recommandation s'effectuera grâce aux communautés d'utilisateurs découvertes à travers les liens multiples qui existent entre eux, notamment via les requêtes qu'ils soumettent à l'entrepôt de données. Le deuxième type de règles servira à l'enrichissement des requêtes par la recommandation d'ajout d'attributs aux requêtes des utilisateurs.

1.1 Exemple illustratif

Dans cette section, nous illustrons notre processus de personnalisation, à travers un exemple simplifié, issu d'un entrepôt de données *PUBS*.

PUBS concerne l'analyse du chiffre d'affaire (*TO*) et de la quantité (*Qty*) de livres vendus. Ces mesures sont observées par rapport aux dimensions suivantes : *Titles* (title_id, title, type, price, advance, notes, pubdate), *Publishers* (pub_id, pub_name, city, state, country), *Stores* (stor_id, stor_name, stor_adress, city, state, zip), *Times* (ord_date, month, year) et *Authors* (au_id, au_lname, au_fname, phone, adress, city, state, zip, contact).

Considérons cinq utilisateurs (U_1, U_2, U_3, U_4, U_5) connectés à *PUBS* et soumettant chacun une séquence différente de requêtes décisionnelles. Une requête décisionnelles est définie de la manière suivante : **Select** a_1, ...,a_n **From** $D_1, D_2, ..., D_n$ **Where** jointures Group By Cube/Rollup/ Grouping Sets a_1, ...,a_n. Nous nous intéressons en particulier aux attributs appartenant à la clause Select car ils définissent l'ensemble des dimensions et des mesures du cube OLAP.

Exemple : L'utilisateur U_4 lance un ensemble de requêtes sur l'entrepôt *PUBS* :

- R_1 = le chiffre d'affaire du magasin *store_400* pour l'année *2013*. Les attributs de PUBS impliqués dans la clause SELECT de R_1 sont (*"TO"*, *"Stores.stor_id", "Times.year"*).
- R_2= le chiffre d'affaire réalisé sur les ventes des ouvrages de type *Computer Science* vendus au niveau des magasins de *Paris* durant *2013*. (*"TO"*, *"Titles.type", "Stores.stor_id", "Times.year")*.
- R_3= le nombre d'exemplaires d'ouvrages écrits par des *auteurs lyonnais* et édités par *Springer* durant l'année *2013*. (*"Qty"*, *"Authors.city", "Publishers.pub_name", "Times.year"*).

De cette façon, les autres utilisateurs formulent d'autres séquences de requêtes d'analyse qui impliquent les attributs déjà exprimés dans les requêtes de U_4. Prenons par exemple, la requête formulée par l'utilisateur U_5 :

- R_1 = le chiffre d'affaire(*TO*) du magasin *store_500* dans la ville de *Washington* par mois. R_1 (*"TO"*, *"Stores.stor_id", "Stores.city", "Times.month"*).

Les règles triadiques que nous pouvons extraire des requêtes R_1 des utilisateurs U_4 et U_5 sont :

- Une règle entre utilisateurs avec condition des attributs $(U_4 \longrightarrow U_5)_{(TO, Stores.stor\_id)}$.
- Une règle entre attributs de la requête avec comme condition des utilisateurs $(TO \longrightarrow Stores.stor\_id)_{(U_4, U_5)}$.

Dans ce papier, nous proposons une approche de personnalisation de requêtes décisionnelles basée sur ces deux types de règles. L'utilisateur interagira avec l'interface de *P-TRIAR* selon deux scénarii possibles. Dans le premier scénario, l'utilisateur U_4 souhaite réaliser de nouvelles analyses sur l'entrepôt. Il se connecte alors à *P-TRIAR* et interroge le fichier log à partir d'une date donnée et demande toutes les RA triadiques qui existent entre lui et les autres utilisateurs ayant comme condition des attributs, tout en fixant un seuil minimal au support et à la confiance. *P-TRIAR* lui affiche alors toutes les règles qui satisfont ses paramètres. Ensuite, U_4 choisira en fonction des attributs qu'il veut interroger la ou les règles qui lui conviennent. Supposons qu'il choisisse la règle $(U_4 \longrightarrow U_5)_{(TO, Stores.stor\_id)}$, *P-TRIAR* lui recommanderai les requêtes d'analyse les plus fréquentes formulées par l'utilisateur U_5 et ayant parmi leurs attributs TO et $Stores.stor\_id$. Contrairement à la règle dyadique $(U_4 \longrightarrow U_5)$ qui recommanderait l'ensemble des requêtes formulées par U_5, les règles triadiques ajoutent une condition sur l'utilisation des attributs de la requête, ainsi la règle est enrichie et le nombre de requêtes à recommander est réduit considérablement. Par conséquent, U_4 pourra choisir parmi ces requêtes celle qui convient le mieux à ses besoins d'analyse avec la possibilité de la modifier en partie. Dans le second scénario, U_4 décide de formuler une requête sur des attributs de l'entrepôt en exploitant les règles qui existent entre attributs des requêtes ayant comme condition des utilisateurs. Il fixe alors les paramètres initiaux, la date, le seuil minimal du support et de la confiance. Ensuite, U_4 choisira les attributs qu'il veut formuler dans sa requête et *P-TRIAR* lui proposera les RA triadiques qui leur sont associés. Supposons, qu'il choisisse l'attribut TO, *P-TRIAR* lui recommande alors l'attribut $Stores.stor\_id$ en se basant sur la règle $(TO \longrightarrow Stores.stor\_id)_{(U_4, U_5)}$. Contrairement à la règle $(TO \longrightarrow Stores.stor\_id)$ qui serait proposée à l'ensemble des utilisateurs dans le cas des règles dyadiques, cette règle sera uniquement recommandée aux utilisateurs U_4 et U_5.

La suite de l'article est organisée de la manière suivante. La section 2 présente un état de l'art sur la personnalisation et la production de règles d'association à partir de données multidimensionnelles. La section 3 présente la modélisation des données issues des fichiers

log de requêtes à travers l'Analyse Formelle de Concepts (AFC) et l'Analyse Triadique de Concepts (ATC). La section 4 décrit notre méthode pour la production de RA triadiques et les algorithmes qui y sont associés. La section 5 décrit le processus de personnalisation de requêtes décisionnelles que nous proposons. Des expérimentations sont effectuées en section 6 pour illustrer la compacité des règles triadiques par rapport aux règles dyadiques et leur apport à la personnalisation. Pour terminer, nous concluons nos travaux et présentons quelques perspectives de recherche en section 7.

2 Etat de l'art

L'état de l'art présenté ici s'articule autour de deux axes de recherche. Le premier définit les travaux ayant trait à la personnalisation dans les systèmes OLAP et le second recense les travaux relatifs à l'ATC et aux différentes approches d'extraction de RA triadiques.

La personnalisation de requêtes a fait l'objet de plusieurs travaux (Bellatreche et al. (2005) ; Adomavicius et Tuzhilin (2005) ; Bentayeb et al. (2009) ; Bentayeb (2011)). Elle a pour vocation d'aider l'utilisateur en se basant généralement sur son comportement et sur ses requêtes précédentes ou celles d'autres utilisateurs. Dans les domaines des bases de données et des entrepôts de données, les différentes techniques de personnalisation ont été classées selon trois catégories (Adomavicius et Tuzhilin (2005) ; Bentayeb (2011)) : les techniques collaboratives (Chatzopoulou et al. (2009) et Golfarelli et al. (2011)) qui exploitent la similarité qui existe entre les profils des utilisateurs et celui pour lequel la recommandation est déterminé ; les techniques basées sur le contenu (Khemiri et Bentayeb, 2012) visent à recommander à un utilisateur des attributs qu'il sollicite fréquemment ; et enfin les techniques hybrides (Stefanidis et al., 2009) qui combinent les deux techniques précédentes. Dans la littérature, les systèmes de recommandation ont comme sources de données des profils utilisateurs, des fichiers log qui sont une historisation structurée des requêtes de chaque utilisateur ou bien des sources externes telles que les ontologies, les pages web, etc.

Plusieurs travaux ont exploité l'idée d'extraction de motifs (Khemiri et Bentayeb, 2013) et de règles d'association (Veloso et al., 2008), à partir des fichiers log, pour la recommandation. Cependant, leurs travaux se sont restreints à un cadre bidimensionnel. Ils représentent les données des fichiers log à travers des matrices d'appariement ($utilisateurs\times$ *requêtes*) ou bien ($attributs\times$ *requêtes*) pour l'extraction de règles d'association ou de motifs. Cette modélisation ne tient pas compte de l'aspect tridimensionnel de ces données. Dans les entrepôts de données, les RA et les motifs qu'ils obtiennent sont très nombreux et de type dyadique. Ce très grand nombre de RA et de motifs rend la tâche de recommandation plus compliquée et ne prend pas en considération les trois ensembles en mêmes temps.

Par ailleurs, l'AFC Wille (1982), Ganter et Wille (1999) et les treillis de Galois constituent une base théorique pour la résolution de nombreux problèmes dans les domaines de l'intelligence artificielle, du génie logiciel et des bases de données. L'ATC a été initialement introduite par Wille (1995) et Lehmann et Wille (1995). Leurs travaux portent sur l'analyse des contextes, des concepts et des treillis de concepts triadiques appelés $trilattices$. Ils définissent de la sorte, la base théorique pour l'ATC. Biedermann (1997) propose un formalisme d'écriture des implications triadiques et Voutsadakis (2002) définit l'analyse de concepts polyadiques et généralise les travaux de Wille (1995) aux contextes formels polyadiques pour produire des concepts formels polyadiques et des n-treillis. Des travaux plus récents liés à l'ATC existent, Ganter et

Obiedkov (2004) considèrent différents types d'implications triadiques dites fortes en partant du formalisme énoncé par Biedermann (1997). Nguyen et al. (2010) proposent une approche de découverte de règles appliquée à des graphes relationnels dynamiques qui peuvent être codés dans des relations n-aires ($n \geq 3$). Les travaux de Missaoui et Kwuida (2011) proposent non seulement une approche de production des RA triadiques mais également des procédures de détermination des concepts et générateurs triadiques à partir des concepts et générateurs dyadiques. Dans Trabelsi (2012), l'auteur traite du calcul des générateurs et des RA triadiques. Toutefois, l'auteur fournie une nouvelle définition des RA triadiques qui est différente de celle de Missaoui et Kwuida (2011) qui, quant à elle, se base sur la définition de *Biedermann*. Dans Cerf et al. (2013), les auteurs proposent la généralisation du concept de RA dans un contexte multidimensionnel en travaillant non plus sur des matrices booléennes mais sur des tenseurs booléens d'arité arbitraire. Ils proposent aussi des mesures de fréquence et de confiance pour définir la sémantique de telles règles.

D'après l'étude bibliographique que nous avons menée, notre travail est le premier à modéliser les données issus des fichiers log à travers un contexte triadique. Notre approche fournit une forme de personnalisation à partir de RA triadique. Nous montrons, à travers notre démarche, comment obtenir des RA triadiques, à partir de ces contextes triadiques, en exploitant uniquement les RA dyadiques sans calculer les concepts et les générateurs triadiques comme le proposent les auteurs cités précédemment. Les RA dyadiques que nous exploitons sont obtenues par l'algorithme de Pasquier (2000) qui est basé sur l'AFC.

3 Modélisation des données log par l'AFC

Dans cette section, nous développons le processus de modélisation des données par l'AFC, collectées implicitement à partir des fichiers log de requêtes des serveurs *OLAP*. Nous nous intéressons, dans ce travail, particulièrement à trois données contenues dans un fichier log de *SQL server* de requêtes à savoir *MSOLAP_User* qui recense les utilisateurs, *Dataset* qui contient les requêtes ainsi que leurs attributs et *StartTime* contenant la date et l'heure du lancement de la requête qui sera utilisé pour déterminer à partir de quelle date sera exploité le fichier log. Ces trois données sont facilement accessibles dans les log de requêtes des entrepôts de données contrairement aux données sur les profils des utilisateurs qui sont souvent cachés à cause de leur aspect privé. Nos définitions se basent sur celles introduites dans (Lehmann et Wille, 1995) d'un contexte triadique et de son équivalent.

Définition 3.1 : (Contexte triadique) En analyse formelle de concepts, un *contexte triadique* est un quadruplet $\mathbb{K} := (R, U, A, Y)$ où R, U, A et Y.

- R, U, A définissent respectivement des **R**equêtes, des **U**tilisateurs et des **A**ttributs (descripteurs et mesures) de la clause SELECT des requêtes.
- $Y \subseteq R \times U \times A$ représente une relation ternaire, où chaque $y \subseteq Y$ représente un triplet : $y = \{(r, u, a) | r \in R, u \in U, a \in A\}$. Autrement dit, une requête *r* est lancée par un utilisateur **u** et qui implique l'attribut a.

Nous illustrons, à travers un exemple (Tableau 1 (a)), le passage des données issues des log vers un contexte triadique. Chaque utilisateur dans U (U_1, U_2,..., U_4) effectue des analyses en lançant une séquence de requêtes notée R (R_1, R_2,..., R_4) où chaque requête est composée d'un ensemble A d'attributs des différentes dimensions et faits de l'entrepôt de données (a_1, a_2,...,a_5). Par exemple, la valeur $a_1a_2a_4$ se trouvant à l'intersection de la première colonne et

$\mathbb{K}$	U_1	U_2	U_3	U_4
R_1	$a_1a_2a_4$	$a_1a_2a_4a_5$	a_1a_3	a_1a_5
R_2	$a_1a_4a_5$	$a_2a_3a_4$	$a_1a_2a_4a_5$	a_4a_5
R_3	$a_1a_2a_4$	a_4a_5	a_1a_2	a_1a_5
R_4	$a_1a_2a_4a_5$	a_2a_4	a_1a_2	a_4a_5
R_5	$a_1a_4a_5$	$a_1a_4a_5$	$a_1a_2a_4a_5$	a_1a_5

(a)

$\mathbb{K}^{(1)}$	U_1					U_2					U_3					U_4				
	a_1	a_2	a_3	a_4	a_5	a_1	a_2	a_3	a_4	a_5	a_1	a_2	a_3	a_4	a_5	a_1	a_2	a_3	a_4	a_5
R_1	1	1		1		1	1		1	1	1		1			1				1
R_2	1			1	1		1	1	1		1	1		1	1				1	1
R_3	1	1		1					1	1	1	1				1				1
R_4	1	1		1	1		1		1		1	1							1	1
R_5	1			1	1	1			1	1	1	1			1	1				1

(b)

FIG. 1 – ***(a)*** *Contexte triadique* $\mathbb{K} := (R, U, A, Y)$*, formé de* $R = \{R_1, R_2, R_3, R_4, R_5\}$ *(requêtes),* $U = \{U_1, U_2, U_3, U_4\}$ *(utilisateurs) et* $A = \{a_1, a_2, a_3, a_4, a_5\}$ *(attributs).* ***(b)*** *Contexte dyadique équivalent* $\mathbb{K}^{(1)}$ *obtenu à partir de* $\mathbb{K}$*.*

la première ligne signifie que l'utilisateur U_1 lance la requête R_1 composée des attributs a_1, a_2 et a_4.

Définition 3.2 : (Contexte dyadique), Un contexte formel dyadique est un triplet $\mathbb{K}^{(1)} := (G, M, I)$ où G est un ensemble d'objets, M un ensemble d'attributs et I une relation binaire entre G et M. Le contexte dyadique obtenu après aplatissement du contexte triadique que nous avons défini (cf. Définition 3.1) est formé de $G = R$ et $M = U \times A$.

Le tableau 1 **(b)** représente le contexte dyadique $\mathbb{K}^{(1)}$ obtenu à partir du contexte triadique $\mathbb{K}$ ainsi : $\mathbb{K}^{(1)} := (R, U \times A, Y^{(1)})$ avec ($(a_i, (a_j, a_k)) \in Y^{(1)} \iff (a_i, a_j, a_k) \in Y$). La valeur 1 pour la première ligne et la première colonne 1 signifie que l'utilisateur U_1 lance la requête R_1 qui implique l'attribut a_1.

Dans ce qui suit, la paire $(a_j, a_k) \in U \times A$ sera notée d'une manière simplifiée par a_j-a_k.

Définition 3.3 (Dérivation) Pour $A \subseteq G$ et $B \subseteq M$, deux sous-ensembles $A' \subseteq M$ et $B' \subseteq G$ sont définis respectivement comme un ensemble d'attributs communs aux objets dans A et un ensemble d'objets qui partagent tous les attributs dans B. Formellement, la dérivation notée $'$ est définie comme suit :

$A' := \{a \in M \mid o\mathrm{I}a \; \forall o \in A\}$ et $B' := \{o \in G \mid o\mathrm{I}a \; \forall a \in B\}$.

Cette proposition définit une paire de correspondance $('\,,'\,)$ entre l'ensemble des parties de G et l'ensemble des parties de M représentant une correspondance de Galois. Les opérateurs de fermeture dans G et M sont notés par $''$. Par exemple, la fermeture de $U_2 - a_4$ est donnée par : $(U_2 - a_4)'' = ((U_2 - a_4)')' = \{R_1, R_2, R_3, R_4, R_5\}' = \{U_1 - a_1, U_1 - a_4, U_2 - a_4, U_3 - a_1, U_4 - a_5\}$.

Définition 3.4 : (Concept formel) Un concept formel (cf) est une paire (A, B) avec $A \subseteq G$, $B \subseteq M$, $A = B'$ et $B = A'$. L'ensemble A, qu'on notera Ext(cf), est appelé *extension* de cf tandis que B est son *intention*, qu'on note Int(cf). Un concept (dyadique) formel correspond à un rectangle maximal dans un contexte dyadique.

Exemple : Comme $\{R_1, R_2, R_3, R_4, R_5\}'$= $\{U_1 - a_1, U_1 - a_4, U_2 - a_4, U_3 - a_1, U_4 - a_5\}$ et $\{U_1 - a_1, U_1 - a_4, U_2 - a_4, U_3 - a_1, U_4 - a_5\}'$ = $\{R_1, R_2, R_3, R_4, R_5\}$, le couple($\{R_1, R_2, R_3, R_4, R_5\}$,$\{U_1 - a_1, U_1 - a_4, U_2 - a_4, U_3 - a_1, U_4 - a_5\}$) forme un concept formel.

Définition 3.5 : (Treillis de concepts) L'ensemble $\mathfrak{B}(\mathbb{K})$ de tous les concepts du contexte $\mathbb{K}$ ordonnés partiellement par : $(X_1, Y_1) \leq (X_2, Y_2) \Leftrightarrow X_1 \subseteq X_2$ constitue un treillis complet, appelé treillis de concepts (Galois) de $\mathbb{K}$ et noté $\underline{\mathfrak{B}}(\mathbb{K})$. La figure 2 montre $\underline{\mathfrak{B}}(\mathbb{K}^{(1)})$ le treillis de concepts obtenu à partir du contexte dyadique $\mathbb{K}^{(1)}$ du tableau 1(b). L'étiquetage du treillis de la figure 2 est réduit au niveau des attributs de sorte que l'intention d'un concept (nœud) n

est donnée par l'union des attributs apparaissant dans le nœud n ainsi que ceux apparaissant dans les concepts qui sont plus petits que n.

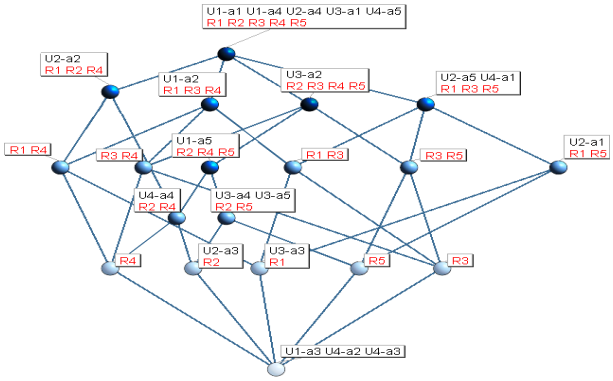

FIG. 2 – *Treillis de Concepts généré à partir du contexte dyadique.*

Définition 3.6 : (Règles d'association dyadique) Soit (G, M, I) un contexte formel dyadique. Une règle d'association (*R*) est de la forme $R : B \rightarrow C\ (s, c)$ où $B, C \subseteq M$ avec $B \cap C = \emptyset$. Le paramètre *s* est donné par : $Supp(R) = \frac{|B' \cap C'|}{|G|}$ est appelé support de la règle R tandis que le paramètre *c* est donné par : $Conf(R) = \frac{|B' \cap C'|}{|B'|}$ est sa confiance Agrawal et al. (1994). Une implication est une RA dont la confiance est égale à 1.

Dans la section qui suit nous exploiterons les RA dyadiques que nous produirons à partir de notre contexte $(\mathbb{K}^{(1)})$ pour générer des RA triadiques.

4 Extraction des Règles d'Association Triadiques

4.1 Définitions

Il ressort de l'étude bibliographique menée jusqu'ici que Biedermann (1997) est le premier à avoir étudié le problème d'extraction d'implications dans des contextes triadiques. Une *implication triadique* a la forme suivante : $(A \rightarrow D)_C$. Cette implication est vraie si "*chaque fois que A est vrai sous toutes les conditions dans C, alors D est aussi vrai sous toutes ces conditions*". Plus tard, Ganter et Obiedkov (2004) ont étendu le travail de Biedermann et ont défini trois types d'implications : Les implications attributs - conditions, les implications attributs conditionnels, les implications conditions attributionnelles. Missaoui et Kwuida (2011) étendent ces définitions aux RA et proposent trois types de règles d'association : les RA (dyadiques) attributs - conditions (*Attributes-Conditions Association Rules* (**ACARs**)) ; les RA (triadiques) attributs conditionnels (*Conditional Attribute Association Rules* (**CAARs**)) ; les RA (triadiques) conditions attributionnelles (*Attributional Condition Association Rules* (**ACARs**)).

Dans ce qui suit, nous considérons le contexte triadique $\mathbb{K} := (R, U, A, Y)$ et son contexte dyadique $\mathbb{K}^{(1)} := (R, U \times A, Y^{(1)})$ de notre exemple (Tableau 1) pour définir les différents types de RA.

Définition 4.1.1 : Une RA *attribut - condition* (**RAA-C**) est une RA dyadique de la forme $A \rightarrow D\ (s, c)$, où A et D sont des sous-ensembles de $U \times A$, s et c représentent respectivement le support et la confiance. Ces RA dyadiques sont extraites à partir du contexte dyadique $\mathbb{K}^{(1)}$.

Exemple : $U_2 - a_1 \rightarrow U_2 - a_5, U_2 - a_4, U_3 - a_1, U_1 - a_1, U_1 - a_4, U_4 - a_1, U_4 - a_5$ $(0.4, 1)$ est une RAA-C de support égale à 40% et de confiance égale à 100%.

Définition 4.1.2 : Une RA attribut conditionnel selon le formalisme de Biedermann (**BCAAR**) est une RA triadique à la notation suivante : $(A \rightarrow D)_C\ (s, c)$, où A et D sont des sous-ensembles de U, et C un sous-ensemble de A et veut dire que *A implique D sous toutes les conditions dans C avec un support s et une confiance c.*

Exemple : la règle $(U_2 \rightarrow U_1)_{a_1 a_2}(0.2, 1)$ est une *BCAAR* avec un support de 20% et une confiance de 100%.

Définition 4.1.3 : Une RA condition attributionnelle (**BACAR**) est une RA triadique à la notation suivante $(A \rightarrow D)_C\ (s, c)$, où A et D sont des sous-ensembles de A, et C est un sous-ensemble de U et veut dire que *A implique D sous toutes les conditions dans C avec un support s et une confiance c.*

Exemple : la règle $(a_2 \rightarrow a_4)_{U_2 U_1}(0.4, 1)$ est une *BACAR* avec un support de 40% et une confiance de 100%.

Ces deux types de RA triadiques (i.e., *BCAAR* et *BACAR*) ont la même notation mais les ensembles des parties gauches, droites et conditions diffèrent.

4.2 Approche proposée

Dans ce qui suit, nous présentons notre démarche en se basant sur des définitions formelles et en l'illustrant par des exemples. Plusieurs approches de recherche et d'analyse de concepts triadiques sont apparues dans la littérature Trabelsi (2012) et Cerf et al. (2013) pour $n = 3$. Missaoui et Kwuida (2011), proposent une approche efficace basée sur l'analyse de contextes triadiques pour l'extraction de RA triadiques. Elle consiste à prendre en entrée un contexte formel triadique qui est aplatit pour produire un contexte dyadique. Ensuite, les concepts dyadiques et les générateurs dyadiques sont générés, puis leur succession est ordonnée pour pouvoir construire le treillis de Galois correspondant. Les concepts triadiques sont alors générés à partir de concepts dyadiques et les générateurs triadiques à partir de générateurs dyadiques. Une fois ces deux ensembles rassemblés, il est alors possible d'extraire les RA triadiques. Ces opérations donnent de bons résultats mais peuvent être éviter par notre alternative qui ne calcul pas ces deux ensembles.

Notre approche se base sur le même fondement théorique que l'approche proposée par Missaoui et Kwuida (2011). Néanmoins, notre démarche d'extraction est différente en termes de données en entrée et nos algorithmes sont appliqués plutôt sur un ensemble de RA dyadiques de type **RAA-C**. Pour extraire ces **RAA-C** nous appliquons l'algorithme de Pasquier (2000) sur le contexte dyadique $\mathbb{K}^{(1)} := (R, U \times A, Y^{(1)})$ obtenu de la projection de l'ensemble des propriétés U sur l'ensemble des conditions A du contexte formel triadique $\mathbb{K} := (R, U, A, Y)$. Ensuite, à partir de ces dernières et des définitions rappelées dans la section 4.1, nous appliquons les algorithmes que nous avons proposés pour la recherche des RA triadiques dans leurs différentes formes : les RA attribut conditionnel à la *Biedermann* (*BCAAR*) et les RA condition attributionnelle à la *Biedermann* (*BACAR*).

4.3 Algorithmes proposés

Le passage des RA dyadiques à l'ensemble des RA et des implications triadiques, dans leurs différentes formes, s'effectue à l'aide d'une procédure principale appelée *TRIAR* . *TRIAR* permet de produire les RA triadiques en faisant appel à deux procédures secondaires *BCAAR* et *BACAR*. Ce choix de décomposition des algorithmes est motivé par la parallélisation de ces deux procédures lors de l'implémentation pour avoir deux types de personnalisation.

Algorithme 1 : Procédure principale. Calcul des RA triadiques.

1: **Procedure** *TRIAR*(D)
2: **In :** $D = \{(LHS, RHS, s, c)\}$
3: **Out :** $\Sigma = \{(L, R, C, t, s, c)\}$
4: $\Sigma \leftarrow \emptyset$;
5: **for** RL = (LHS, RHS, s, c) dans D **do**
6: {A_L and M_L stockent les attributs et les conditions resp. de la prémisse LHS de la règle dyadique courante RL}
7: $A_L \leftarrow$DISTINCT A(LHS)
8: $M_L \leftarrow$DISTINCT M(LHS)
9: **if** $Size(A_L) \times Size(M_L) = Size(LHS)$ **then**
10: $\Sigma \leftarrow \Sigma \cup \{(\text{BCAARs}(A_L, M_L, RHS), 1, s, c)\}$
11: $\Sigma \leftarrow \Sigma \cup \{(\text{BACARs}(A_L, M_L, RHS), 2, s, c)\}$
12: **out** Σ

La procédure principale *TRIAR* (Algorithme 1) se compose de trois parties. La première partie (lignes 4-9) correspond à une procédure de tri qui permet de déterminer si une RA dyadique est éligible pour se transformer en une RA triadique ou non. Nous collectons les valeurs distinctes des attributs dans l'ensemble A_L (ligne 7) de la prémisse de la règle LHS, les valeurs distinctes des conditions dans l'ensemble M_L (ligne 8) de LHS ; et nous vérifions si le produit de ces valeurs correspond à la taille de LHS (ligne 9). Les deux autres parties (lignes 10 et 11) correspondent aux procédures *BCAAR* et *BACAR* (Algorithmes 2 et 3) qui permettent de produire l'ensemble des RA triadiques. Nous avons en entrée de *TRIAR* un ensemble de RA dyadique **(RAA-C)** (D) où chaque règle se présente sous la forme suivante (LHS, RHS, s, c) représentant respectivement (la prémisse de la règle, la conclusion de la règle, le support et la confiance).

Exemple : la règle $U_3 - a_4, U_4 - a_4 \rightarrow U_2 - a_3, U_2 - a_2, U_2 - a_4, U_3 - a_1, U_3 - a_5, U_3 - a_2, U_1 - a_1, U_1 - a_5, U_1 - a_4, U_4 - a_5$(sup = 0.20 ; conf = 1.00) aura la forme suivante ($\{U_3 - a_4, U_4 - a_4\}$, $\{U_2 - a_3, U_2 - a_2, U_2 - a_4, U_3 - a_1, U_3 - a_5, U_3 - a_2, U_1 - a_1, U_1 - a_5, U_1 - a_4, U_4 - a_5\}$, 0.20,1).

En sortie de la procédure *TRIAR*, nous avons un ensemble de RA triadiques (Σ), où chaque règle se présente sous la forme suivante (L, R, C, t, s, c), représentant respectivement la prémisse de la règle, la conclusion de la règle, la condition de la règle, le type de la règle (1 : *BCAAR* ; 2 : *BACAR*), le support et la confiance.

Exemple : la *BCAAR* $(U_3U_4 \rightarrow U_2U_1)_{a_4}$ (sup = 0.20 ; conf = 1.0)) aura la forme suivante $(U_3U_4, U_2U_1, a_4, 1, 0.20, 1.0)$.

Pour dérouler notre algorithme, nous prenons à titre d'exemple la règle dyadique ($\{U_3 - a_4, U_4 - a_4\}$, $\{U_2 - a_3, U_2 - a_2, U_2 - a_4, U_3 - a_1, U_3 - a_5, U_3 - a_2, U_1 - a_1, U_1 - a_5, U_1 -$

$a_4, U_4 - a_5\}, 0.20, 1)$. Des lignes 5 à 8 de l'algorithme 1, nous créons les deux ensembles A_L et M_L qui contiennent respectivement les attributs distincts et les conditions distinctes de la prémisse de la règle LHS $\{U_3 - a_4, U_4 - a_4\}$. Par conséquent, A_L= $\{U_3, U_4\}$, M_L=$\{a_4\}$. Ceci implique que le produit $Size(A_L) \times Size(M_L)$ = 2 (ligne 9) soit égal à $Size(LHS)$, comme la partie M_L va devenir une condition pour les règles construites. Tous les éléments de A_L doivent vérifier cette condition ainsi cette règle est éligible à devenir une RA triadique. Les lignes 10 et 11 de l'algorithme 1 font appel aux deux procédures *BCAAR* et *BACAR* pour produire les deux types de règles triadiques.

La procédure $BCAARs$ (Algorithme 2) prend en entrée trois ensembles A_L, M_L et RHS. L'ensemble M_L représente les conditions qui s'appliquent à tous les attributs dans l'ensemble A_L et nous voulons trouver dans RHS d'autres attributs qui sont affectés par les mêmes conditions, d'où la recherche des conditions des lignes 6-8. Ces attributs seront isolés dans la ligne 10 ($group\ by$ sur les atttributs), pour permettre de voir si leurs conditions matchent les conditions de M_L (ligne 12), si elles sont identiques à celles de M_L nous pouvons construire une règle.

Algorithme 2 : Calcul des RA triadiques *BCAAR*, de la forme : $(A \rightarrow D)_C$

1: **Procedure** BCAARs(A_L, M_L, RHS)
2: **In :** A_L, M_L, RHS
3: **Out :** *BCAAR* = (A_L, A_R, M_L)
4: $A_R \leftarrow \emptyset$; $Temp \leftarrow \emptyset$
5: **for** $e \in RHS$ **do**
6: **if** MODUS$(e) \in M_L$ **then**
7: {MODUS(e) est la condition de l'élément dans RHS}
8: $Temp \leftarrow Temp \cup \{e\}$
9: **if** $Temp \neq \emptyset$ **then**
10: Grouper les éléments de $Temp$ ayant la même partie d'attributs en commun dans un conteneur B = $b_1, ..., b_n$
11: **for** $elem \in B$ **do**
12: **if** $Size(elem) = Size(M_L)$ **then**
13: $A_R \leftarrow A_R \cup \{Attr(elem)\}$
14: **if** $A_R \neq \emptyset$ **then**
15: **out** (A_L, A_R, M_L)

Le déroulement de l'algorithme s'effectue comme suit : après l'initialisation des paramètres (lignes 2-4), prendre la conclusion de la règle RHS (ligne 5) qui correspond à $\{U_2 - a_3, U_2 - a_2, U_2 - a_4, U_3 - a_1, U_3 - a_5, U_3 - a_2, U_1 - a_1, U_1 - a_5, U_1 - a_4, U_4 - a_5\}$ dans l'exemple, et calculer le $Modus$ de chaque élément qui correspond à la condition. Pour le premier élément $Modus(U_2 - a_3)$=$\{a_3\}$ le test montre qu'il n'est pas inclus dans l'ensemble M_L = $\{a_4\}$ condition non vérifiée, la boucle passe à l'élément suivant. Pour le quatrième élément $Modus(U_2 - a_4)$=$\{a_4\}$ il est inclus dans M_L condition vérifiée, La variable *Temp* reçoit cet élément $(U_2 - a_4)$, ensuite à ligne 10,nous groupons dans un conteneur noté B les éléments qui ont la même partie attribut, dans notre exemple $(U_2 - a_4), (U_1 - a_4)$ seront contenus dans (B). L'algorithme 2 vérifie, (lignes 11-12), pour chaque élément contenu dans (B) si la taille de cet élément est égale à la taille de M_L. Dans notre exemple, ces deux entités sont égales pour les deux éléments car ils ont une taille égale à 1. La règle formée du triplet (A_L, A_R, M_L)

= $(\{U_3, U_4\}, \{U_2, U_1\}, a_4)$ est alors constituée, à laquelle, il est rajouté le type, le support et la confiance. Le résultat est alors : *BCAAR* $(U_3U_4 \rightarrow U_2U_1)_{a_4}$, type = 1, Sup = 0.20 et Conf= 1.00. C'est le point de sortie de l'algorithme 2 et la règle est rajoutée à l'ensemble des *BCAAR*.

Algorithme 3 : Calcul des RA triadiques *BACAR*, de la forme : $(A \rightarrow D)_C$

1: **Procedure** BACARs(A_L, M_L, RHS)
2: **In :** A_L, M_L, RHS
3: **Out :** *BACAR*= (M_L, M_R, A_L)
4: $M_R \leftarrow \emptyset$; $Temp \leftarrow \emptyset$
5: **for** $e \in RHS$ **do**
6: **if** ATTRIB$(e) \in A_L$ **then**
7: {ATTRIB(E) est l'attribut de l'élément courant dans la RHS}
8: $Temp \leftarrow Temp \cup \{e\}$
9: **if** $Temp \neq \emptyset$ **then**
10: Grouper les éléments de $Temp$ ayant la même partie condition dans un conteneur $B = b_1, ..., b_n$
11: **for** $elem \in B$ **do**
12: **if** $Size(elem) = Size(A_L)$ **then**
13: $M_R \leftarrow M_R \cup \{Cond(elem)\}$
14: **if** $M_R \neq \emptyset$ **then**
15: **out** (M_L, M_R, A_L)

Dans la procédure $BACARs$ (algorithme 3), nous avons en entrée trois ensembles A_L, M_L et RHS. L'ensemble A_L représente les attributs qui s'appliquent à toutes les conditions dans l'ensemble M_L et nous voulons trouver dans RHS d'autres attributs qui sont affectés par les mêmes conditions, d'où la recherche des conditions des lignes 6-8. Ces attributs seront isolées dans la ligne 10 ($group\ by$ sur les conditions), pour permettre de voir si leurs attributs matchent les attributs de M_L (ligne 12), si les attributs sont identiques à ceux de M_L nous pouvons construire une règle en adoptant la même démarche décrite pour l'algorithme 2.

4.4 Étude de complexité

Dans ce qui suit, nous présentons l'étude de la complexité de notre algorithme principal *TRIAR*. Ce dernier fait appel aux procédures *BCAAR* et *BACAR*. Il prend en entrée un ensemble de RA dyadiques D. Ce dernier est obtenu à partir d'un contexte formel dyadique $\mathbb{K} := (R, U \times A, Y)$. La taille maximale d'une RA dyadique est donnée par $|U| * |A|$. La complexité globale de l'algorithme est linéaire en $|D|$ et s'effectue en $O(|D| * 2(|U| + |A|))$. Cette complexité est obtenue par l'étude de la boucle "pour" (ligne 5), qui parcourt une seule fois l'ensemble des règles D, elle est donnée par : la ligne 7 s'effectue en $O(|U|)$ car au pire, nous pouvons avoir dans une règles l'ensemble des propriétés du contexte ; la ligne 8 s'effectue en $O(|A|)$ car au pire nous pouvons avoir dans une règle l'ensemble des propriétés du contexte ; le test de la ligne 9 s'effectue en $O(|D|)$ car c'est l'ensemble de règles qui est parcouru pour tester leur éligibilité à devenir des règles triadiques ; les instructions 10 et 11, appellent respectivement la procédure *BCAAR* et *BACAR*. Ces appels s'effectuent au pire en $O(|D| * |U| + |A|)$, dans le cas où toutes les RA dyadiques sont éligibles à devenir des RA triadiques.

5 Architecture de *P-TRIAR*

Le processus de personnalisation que nous proposons *P-TRIAR* comporte cinq étapes (cf. Figure 3).

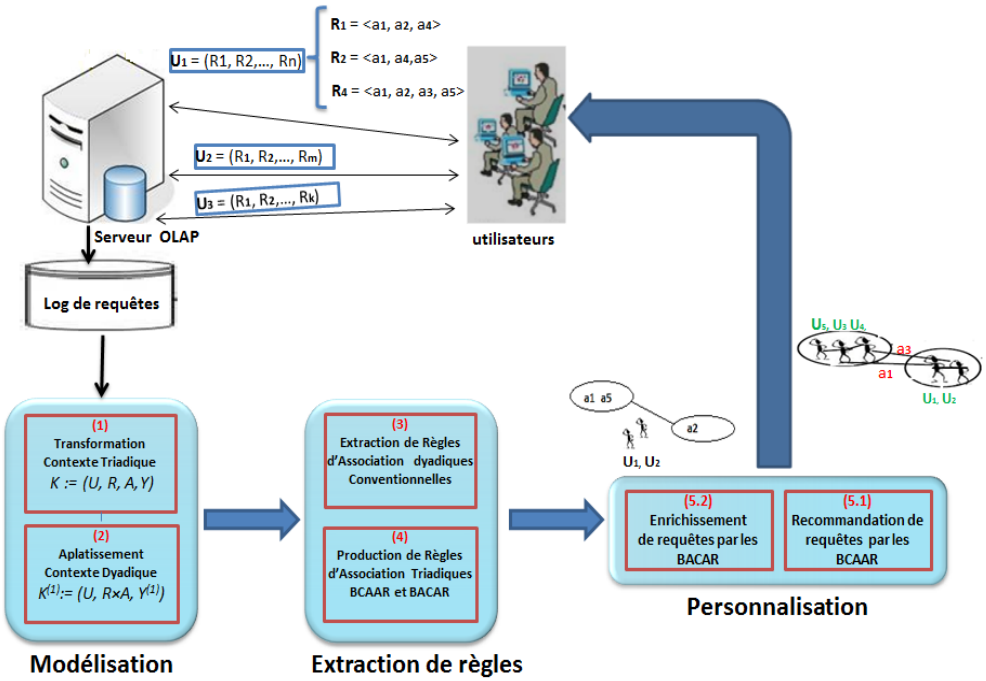

FIG. 3 – *Architecture de P-TRIAR*

Dans la section 3, nous avons décrit les trois premières étapes à savoir : La modélisation par un contexte triadique des données des log de requêtes du serveur d'analyse *OLAP* ; Le passage de ce contexte triadique vers un contexte dyadique et enfin la production de règles d'association dyadiques conventionnelles de type *prémisse* $\rightarrow$ *conclusion*. Ensuite, dans la section 4, nous avons détaillé l'approche que nous proposons pour la génération d'un ensemble de règles d'association triadiques de type $(\textit{prémisse} \rightarrow \textit{conclusion})_{(condition)}$ par *factorisation* des règles dyadiques. Dans ce qui suit, nous décrivons la cinquièmes étape de *P-TRIAR* qui concerne l'exploitation des RA triadiques (*BCAAR* et *BACAR*) obtenues par nos algorithmes.

5.1 Recommandation de requêtes par les *BCAAR*

Les *BCAAR* déterminent les associations qui existent entre utilisateurs ayant comme condition des attributs. Autrement dit, ce type de règles nous permet de découvrir les relations qui existent entre utilisateurs à travers les attributs et mesures impliquées dans leur requêtes. Par exemple la *BCAAR* $(U_4 \rightarrow U_3U_1)_{a_1}(0.6, 1)$ stipule que chaque fois qu'une requête est soumise par l'utilisateur U_4 et qui contient l'attribut a_1, les utilisateurs U_3 et U_1 soumettent une requête qui contient ce même attribut, avec un support de 60% et une confiance de 100%. Cette

règle met en évidence la similarité qui existe entre l'utilisateur U_4 et les utilisateurs U_3 et U_1 mais sous condition d'interroger l'attribut a_1. Nous retrouvons à travers cette règle l'aspect collaboratif car elle permet de former un lien communautaire entre trois utilisateurs. Ce lien communautaire est conditionné par l'implication de l'attribut a_1 dans leurs requêtes et le degré de ce lien a un support et une confiance spécifiques.

Le premier scénario de personnalisation que nous avons décrit dans l'exemple illustratif 1.1 sera basé sur les *BCAAR*. Dans ce scénario, l'utilisateur se connecte définit ces paramètres (la date à partir de laquelle il veut explorer les log, le support minimal et la confiance minimale) et désire connaitre les liens qu'il entretien avec les autres utilisateurs. *P-TRIAR* lui affiche les règles qui satisfont ces paramètres. Supposons que U_4 choisisse la règle de notre exemple, *P-TRIAR* va lui recommander un nombre de requêtes décisionnelles que U_4 désire. Ces requêtes seront filtrées et classées : par fréquences, par utilisateurs (U_3 et U_1) et par attributs (a_1). Et pourra choisir celles qui conviennent à ces besoins d'analyse. Si l'utilisateur désire accéder directement que *P-TRIAR* lui propose des requêtes sans avoir à choisir parmi les $BCAAR$, *P-TRIAR* détecte quel utilisateur est connecté et lui propose un nombre de requêtes filtrés par nombre d'utilisateurs et nombre d'attributs, c'est à dire, en se basant sur les règles qui ont le plus grand nombre d'utilisateurs dans la partie conclusion de la règle et le plus grand nombre d'attributs dans sa partie condition.

5.2 Enrichissement de requêtes par les *BACAR*

Les *BACAR* déterminent les associations qui existent entre attributs ayant comme condition des utilisateurs. Ce type de règles, nous permet de découvrir les relations qui existent entre les attributs (descripteurs et mesures) impliqués dans une requête à travers les utilisateurs qui la formule. Par exemple la *BACAR* $(a_2 \rightarrow a_4)_{U_2U_1}(0.4, 1)$ est vraie lorsque pour chaque fois qu'une requête est soumise et qui implique l'attribut a_2, l'attribut a_4 est impliqué aussi dans la requête à condition que les utilisateurs qui la formule soit U_2 et U_1.

Le second scénario de recommandation décrit dans la section 1.1 sera basé sur les *BACAR*. Dans ce scénario, l'utilisateur définit les mêmes paramètres du premier scénario et désire formuler une requête d'analyse en s'inspirant des liens qui existent entre les attributs de l'entrepôt. Supposons que l'utilisateur U_2 est connecté est choisi la *BACAR* $(a_2 \rightarrow a_4)_{U_2U_1}$ qui veut dire que chaque fois qu'une requête est soumise et qui contient l'attribut a_2, l'attribut a_4 est impliqué aussi dans la requête à condition que les utilisateurs qui la formule soit U_2 et /ou U_1, avec un support de 40% et une confiance de 100%. Cette règle met en évidence la similarité qui existe entre l'attribut a_2 et l'attribut a_4 mais sous condition que l'utilisateur U_1 ou U_2 formule la requête. *P-TRIAR* se base sur cette règle pour enrichir la requête de l'utilisateur U_2 par la recommandation de l'attribut a_4 comme élément de sa requête.

6 Experimentations

Dans un premier temps, pour évaluer et valider notre approche, nous avons choisi un autre jeu de données, en l'occurrence la base de données *mushroom* [1], qui est un jeu de données connu dans le domaine de fouille des RA. Ce choix est motivé par deux propriétés que l'on

1. $http://archive.ics.uci.edu/ml/datasets/Mushroom$

retrouve dans les fichiers log des grands entrepôts de données à savoir la densité et la forte corrélation entre ces éléments. Nous avons prétraité *mushroom* pour revenir à un contexte triadique, l'ensemble des 128 attributs initiaux a été réparti dans 2 sous ensembles de données représentant des utilisateurs et des attributs et chaque tuple représente une requête. Chaque jeu de données que nous avons formé contient chacun 16 utilisateurs et 8 attributs par utilisateur. Le premier sous ensemble de données ($QUART$) contient 2016 requêtes, 16 utilisateurs et 8 attributs ; le deuxième ($DEMI$) contient 4033 requêtes, 16 utilisateurs et 8 attributs ; et enfin le troisième (TOT) contient 8066 requêtes et le même nombre d'utilisateurs et d'attributs.

Pour extraire les RA triadiques, notre approche prend en entrée directement les RA dyadiques sans passer par l'étape de génération de concepts et de générateurs triadiques évitant ainsi des couts supplémentaires. L'ensemble de RA dyadiques en entrée est différent de l'ensemble de concepts et générateurs dyadiques que prennent en entrée les auteurs de Missaoui et Kwuida (2011).

	QUART	DEMI	TOT
RAD	143192	265649	579199
RC	2921	3331	10181
BCAAR	258	265	2377
BACAR	141	152	281

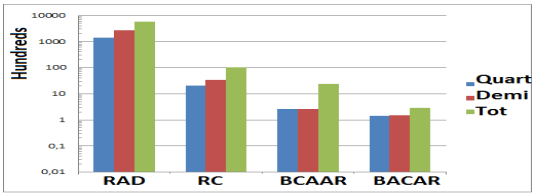

FIG. 4 – *Nombre de RA par type de règles.(échelle* $log10$*)*

La figure 4 représente la variation du nombre de règles obtenues par type de règles et par sous ensemble de données avec un support minimale de 0.1% et une confiance minimale de 0.1%. Pour avoir une idée sur les temps d'exécution, par exemple pour le plus grand jeu de données TOT, 10.181 RA dyadiques candidates (RC) (à partir de 579.199 RA dyadiques) ont été élues pour devenir des RA triadiques et cela en 73 ms seulement. A partir de ces RC, nous avons obtenu en 2 ms, 2.377 *BCAAR* et 281 *BACAR*. Nos résultats montrent que le nombre de RA triadiques qui nous serviront à la personnalisation selon les scénarii décrit dans la section 5, est beaucoup plus petits que les RA dyadiques, en moyenne on obtient 4 RA triadiques de types *BCAAR* et *BACAR* confondues, pour 1000 RA dyadiques. Par rapport aux systèmes qui recommandent des requêtes à partir des RA dyadiques nous aurons un ensemble beaucoup plus compacte de règles auxquels s'ajoute une condition qui enrichi leurs sémantique. Les temps de réponses obtenus permettent une personnalisation rapide aux utilisateurs.

Les tests que nous avons réalisé sur l'entrepôt *PUBS* ont porté sur cinq utilisateurs et 50 requêtes décisionnelles, par utilisateur, formé des 34 attributs distincts que contient *PUBS*. Nous avons obtenu, avec un seuil de support et de confiance minimal de 50%, au total 73 *BCAAR* et 69 *BACAR* à partir de 10.538 RA dyadiques. Par exemple pour l'utilisateur U_4 nous pouvons lui recommander 14 *BCAAR* et 12 *BACAR* selon son choix d'analyse. Les règles que nous avons obtenues ont servi à décrire l'exemple de motivation.

7 Conclusion

Dans cet article, nous avons décrit un nouveau processus de personnalisation d'analyses OLAP, en particulier pour la recommandation collaborative et l'enrichissement de requêtes,

basé sur les fichiers log des requêtes utilisateurs. Nous avons, dans un premier temps, modélisé les données issues des fichiers *log* en utilisant l'analyse formelle de concepts pour construire des contextes triadiques. Ensuite, nous avons proposé une nouvelle méthode qui permet d'exploiter les idées issues de l'analyse triadique de concepts pour générer les RA triadiques, à partir des contextes triadiques, et produire ce type de règles en exploitant seulement des RA dyadiques sans avoir à manipuler des concepts et des générateurs triadiques. Nous avons ainsi pu montrer comment obtenir des RA triadiques de types *BCAAR* et *BACAR* moins nombreuses et plus compactes que les règles dyadiques, tout en véhiculant une sémantique plus riche. Pour valider notre approche de personnalisation de requêtes décisionnelles, nous avons développé *P-TRIAR*, un prototype logiciel permettant d'extraire ces deux types de règles à partir de fichiers log, puis procède soit à la recommandation ou à l'enrichissement de requêtes selon le besoin de l'utilisateur. Ce travail ouvre de nombreuses perspectives de recherche. Nous envisageons à court terme de réaliser un système qui collecte les préférences des utilisateurs à travers leur choix des différentes règles de personnalisation et des requêtes recommandées pour les prendre en considération dans leurs futurs choix. A moyen terme, nous planifions de généraliser les algorithmes proposés à la détection de communautés dans des réseaux sociaux présentant des liens multiples.

8 Remerciements

Nous remercions Rokia Missaoui pour sa collaboration dans ce travail.

Références

Adomavicius, G. et A. Tuzhilin (2005). Toward the next generation of recommender systems : A survey of the state-of-the-art and possible extensions. *Knowledge and Data Engineering, IEEE Transactions on 17*(6), 734–749.

Agrawal, R., R. Srikant, et al. (1994). Fast algorithms for mining association rules. In *Proc. 20th Int. Conf. Very Large Data Bases, VLDB*, Volume 1215, pp. 487–499.

Bellatreche, L., A. Giacometti, P. Marcel, H. Mouloudi, et D. Laurent (2005). A personalization framework for olap queries. In *DOLAP*, pp. 9–18.

Bentayeb, F. (2011). *Entrepôts et analyse en ligne de données complexes centrés utilisateur : un nouveau défi*. Hdr, Université Lumière - Lyon II.

Bentayeb, F., O. Boussaid, C. Favre, F. Ravat, et O. Teste (2009). Personnalisation dans les entrepôts de données : bilan et perspectives. In *EDA*, pp. 7–22.

Biedermann, K. (1997). How triadic diagrams represent conceptual structures. In *ICCS*, pp. 304–317.

Cerf, L., J. Besson, T. K. N. Nguyen, et J.-F. Boulicaut (2013). Closed and Noise-Tolerant Patterrns in N-ary Relations. *Data Mining and Knowledge Discovery 26*(3), 574–619.

Chatzopoulou, G., M. Eirinaki, et N. Polyzotis (2009). Query recommendations for interactive database exploration. In *SSDBM*, pp. 3–18.

Ganter, B. et S. A. Obiedkov (2004). Implications in triadic formal contexts. In *ICCS*, pp. 186–195.

Ganter, B. et R. Wille (1999). *Formal concept analysis - mathematical foundations*. Springer.

Golfarelli, M., S. Rizzi, et P. Biondi (2011). myolap : An approach to express and evaluate olap preferences. *IEEE Trans. Knowl. Data Eng. 23*(7), 1050–1064.

Khemiri, R. et F. Bentayeb (2012). Interactive query recommendation assistant. In *DEXA Workshops*, pp. 93–97.

Khemiri, R. et F. Bentayeb (2013). Fimioqr : Frequent itemsets mining for interactive olap query recommendation. In *DBKDA 2013*, pp. 9–14.

Lehmann, F. et R. Wille (1995). A triadic approach to formal concept analysis. In *ICCS*, pp. 32–43.

Missaoui, R. et L. Kwuida (2011). Mining triadic association rules from ternary relations. In *ICFCA*, pp. 204–218.

Nguyen, K. N. T., L. Cerf, et J.-F. Boulicaut (2010). Sémantiques et calculs de règles descriptives dans une relation n-aire. In *BDA'10*, pp. 1–20.

Pasquier, N. (2000). *Data Mining : algorithmes d'extraction et de réduction des règles d'association dans les bases de données*. Ph. D. thesis.

Stefanidis, K., M. Drosou, et E. Pitoura (2009). You may also like results in relational databases. In *PersDB 2009*, pp. 37–42.

Trabelsi, C. (2012). *Contributions à la Fouille et à la Recherche d'Information dans les Folksonomies*. Ph. D. thesis, Université de Tunis El Manar.

Veloso, A., H. M. de Almeida, M. A. Gonçalves, et W. M. Jr. (2008). Learning to rank at query-time using association rules. In *SIGIR*, pp. 267–274.

Voutsadakis, G. (2002). Polyadic concept analysis. *Order 19*(3), 295–304.

Wille, R. (1982). Restructuring lattice theory : An approach based on hierarchies of concepts. In I. Rival (Ed.), *Ordered Sets*, pp. 445–470.

Wille, R. (1995). The basic theorem of triadic concept analysis. *Order 12*(2), 149–158.

Summary

This article describes a new personalization process for decision-support queries through a new approach of triadic association rules mining. This process uses the log files of users and has five steps: (1) generation of a triadic context from the log files of an OLAP query analysis server (2) mapping of a dyadic context into a triadic one; (3) computation of (conventional) dyadic association rules; (4) generation of triadic association rules through a *factorization* of dyadic ones. The advantage of the former rules is that they are less numerous and more compact than dyadic rules and they convey a richer semantics. And finally, (5) exploitation of these triadic rules for personalization. To validate our approach we developed a personalization software prototype *P-TRIAR* (OLAP Personalization based on TRIadic Association Rules) to extract triadic rules from query log files.