Résumés et interrogations de logs de requêtes OLAP

Résumé. Une façon d'assister l'analyse d'entrepôt de données repose sur l'exploitation et la fouille de fichiers logs de requêtes OLAP. Mais, à notre connaissance, il n'existe pas de méthode permettant d'obtenir une représentation d'un tel log qui soit à la fois concise et exploitable. Dans ce papier, nous proposons une méthode pour résumer et interroger des logs de requêtes OLAP. L'idée de base est qu'une requête résume une autre requête et qu'un log, qui est une séquence de requêtes, résume un autre log. Notre cadre formel est composé d'une algèbre simple destinée à résumer des requêtes OLAP, et d'une mesure évaluant la qualité du résumé obtenu. Nous proposons également plusieurs stratégies pour calculer automatiquement des résumés de logs de bonne qualité, et nous montrons comment des propriétés simples sur les résumés peuvent être utilisées pour interroger un log efficacement. Des tests sur des logs de requêtes MDX ont montré l'intérêt de notre approche.

1 Introduction

Il est maintenant établi que l'exploitation des logs de requêtes peut aider l'utilisateur qui analyse de grandes bases de données ou des entrepôts de données Khoussainova et al. (2009). C'est particulièrement le cas dans un contexte collaboratif, par exemple pour proposer des recommandations Chatzopoulou et al. (2009); Giacometti et al. (2009a,b); Stefanidis et al. (2009). Mais, à notre connaissance, même le simple fait de fournir à l'utilisateur une représentation concise, i.e. un résumé, de ce qui apparaît dans un log de grande taille n'a que très rarement été abordé dans la littérature Khoussainova et al. (2009). Or un résumé aurait de nombreux avantages, comme aider l'administrateur à paramétrer le serveur OLAP s'il indique les membres fréquemment accédés, permettre à un utilisateur d'avoir une première idée du genre de requêtes lancées par d'autres utilisateurs, ou aider l'utilisateur à lancer de nouvelles sessions d'analyse, par exemple en ne refaisant pas ce qui a déjà été fait, ou à écrire de nouvelles requêtes, en partant de, ou en prenant exemple sur, ce qui a déjà été fait.

Dans cet article, nous développons les travaux entrepris sur le résumé de log de requêtes OLAP XXX (XXXa), où nous proposions un langage de manipulation de requêtes (appelé *QSL*) pour résumer des requêtes OLAP, une mesure de la qualité d'un résumé et un algorithme glouton de construction automatique de résumés de log de requêtes utilisant *QSL*. Sur cette base, nous

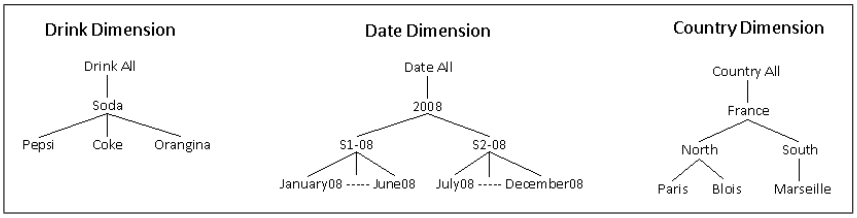


FIG. 1 – *Les dimensions de notre exemple jouet*

proposons un nouvel algorithme de calcul de résumés et deux sous-langages de *QSL* dont nous étudions les propriétés. Nous indiquons comment ces propriétés sont utilisées pour interroger un log. Finalement, nous présentons l'implémentation de notre cadre et quelques tests illustrant son efficacité.

Ce papier est organisé comme suit. La section suivante motive et présente notre approche. Nous rappelons notre cadre dans les sections 3 et 4, en ajoutant les propriétés des opérateurs de *QSL*. Dans la section 5, nous présentons notre nouvel algorithme de résumé et introduisons quelques propriétés des résumés obtenus à l'aide des sous-langages de *QSL*. La section 6 présente l'exploitation des propriétés de notre cadre pour interroger des logs. La section 7 présente notre implémentation et les tests conduits. Nous discutons des travaux connexes et concluons dans la section 8.

2 Motivation et présentation intuitive

Dans cette section, nous illustrons via un exemple jouet l'intérêt de notre méthode et son fonctionnement. L'exemple se base sur un cube de données décrivant des ventes de boissons, et dont les dimensions sont données Figure 1. Considérons que les requêtes déjà lancées par un utilisateur sur cet entrepôt sont loggées sous forme de la séquence $L = \langle q_1, q_2, q_3 \rangle$ où chaque requête demande respectivement :

q_1 les ventes de Pepsi et Coke pour July 2008, à Paris et Marseille,

q_2 les ventes de Coke pour July 2008, dans les régions North et South,

q_3 les ventes d'Orangina pour le second semestre 2008, dans les régions North et South.

Tout d'abord notons que plusieurs manières de résumer sont pertinentes. Par exemple, en résumant le log par la seule requête mentionnant les membres les plus fréquemment interrogés, ce qui intéressera un administrateur souhaitant savoir quels indexes positionner. Dans notre exemple une telle requête demanderait les ventes de Coke dans les régions North et South pour July 2008 (la requête q_2). Une autre manière de résumer permettrait à un utilisateur de savoir grossièrement ce qui a été fait sur le cube, sous la forme d'une requête interrogeant les ventes de soda en France en 2008. Si l'utilisateur est intéressé par plus de détails, il pourra ensuite interroger le log pour trouver des requêtes précises. Un autre intérêt d'un résumé, plus compact qu'un log, est donc de pouvoir être utilisé en amont du log pour rendre l'interrogation plus efficace.

Dans ce papier, nous proposons que les résumés soient exprimés de manière flexible et déclarative via un langage de manipulation de requêtes appelé *QSL*. Les expressions définies avec cette algèbre permettent de résumer plusieurs requêtes en les combinant sous la forme d'une seule requête. Pour appliquer ce principe à un log, nous proposons un algorithme construisant

des expressions de *QSL* pour résumer des sous-séquences du log. La qualité des résumés ainsi produits est évaluée par le biais d'une mesure étendant les mesures classiques de rappel et précision.

3 Le langage *QSL*

Dans cette section, nous définissons formellement notre langage de résumé de requêtes.

3.1 Définitions préliminaires

Comme notre langage de résumé de requêtes est adapté aux requêtes OLAP, nous commençons par définir une requête OLAP. Notons que dans cet article, nous ne nous intéressons pas au résultat de la requête de sorte que la définition d'un résultat de requête n'est pas donnée. Un cube n-dimensionnel $C = \langle D_1, ..., D_n, F\rangle$ est défini comme les $n+1$ instances de relation d'un schéma en étoile avec une instance de relation pour chacune des n dimensions D_i et une instance de relation pour la table de fait F. Pour une dimension D_i de schéma $S = \{L_1^i, \ldots, L_{d_i}^i\}$, un membre m est une constante de $\bigcup_{L_j^i \in S} \pi_{L_j^i}(D_i)$. Pour une dimension D_i, nous considérons que les membres sont arrangés selon une hiérarchie $<_i$ et nous notons $m <_i m'$ (ou $m < m'$ lu m' couvre m) le fait que le membre m' est un ancêtre de m dans cette hiérarchie. Soit un tel cube, une référence de cellule (ou cellule par abus) est un n-uplet $\langle m_1, ..., m_n\rangle$ où m_i est un membre de la dimension $D_i, \forall i \in [1, n]$. Nous définissons des requêtes multidimensionnelles comme des ensembles de références qui peuvent être exprimés comme des produits cartésiens de multi-ensembles. La raison pour laquelle nous utilisons des multi-ensembles est qu'ainsi nous pouvons définir des opérateurs qui comptent le nombre d'occurrences des membres.

Soit un cube n-dimensionnel $C = \langle D_1, \ldots, D_n, F\rangle$ et R_i un multi-ensemble de membres de dimension $D_i, \forall i \in [1, n]$. Une requête q sur C est un multi-ensemble de références $q = R_1 \times \ldots \times R_n$. Un log L est une séquence de requêtes notée $L = \langle q_1, \ldots, q_m\rangle$. L'ensemble des requêtes apparaissant dans un log L est noté $queries(L)$.

Par la suite, nous considérons un cube n-dimensionnel $C = \langle D_1, \ldots, D_n, F\rangle$. Dans les définitions suivantes, i varie entre 1 et n. Pour une requête q donnée, $m_i(q)$ est son multi-ensemble de membres dans la dimension D_i. Une requête q sera notée $\times_i m_i(q)$ et le multi-ensemble $m_i(q)$ sera noté $\langle S_i, f_i\rangle$ où S_i est un ensemble et f_i une fonction retournant les occurrences de chaque élément de S_i.

Exemple 3.1 *Considérons les trois requêtes q_1, q_2 et q_3 de notre exemple décrit dans la section précédente. Nous avons $m_1(q_1) = \{Pepsi, Coke\}$, $m_2(q_1) = \{July08\}$, $m_3(q_1) = \{Paris, Marseille\}$. Cette requête est l'ensemble de références : $q_1 = \{Pepsi, Coke\} \times \{July08\} \times \{Paris, Marseille\}$. Les ensembles de références de q_2 et q_3 sont : $q_2 = \{Coke\} \times \{July08\} \times \{North, South\}$ et $q_3 = \{Orangina\} \times \{S2\text{-}08\} \times \{North, South\}$.*

3.2 Les opérateurs de *QSL*

QSL est un langage algébrique composé d'opérateurs unaires et binaires qui manipulent des requêtes et retournent une requête, appelée la requête résumée (ou résumé par abus). Ces opérateurs construisent une nouvelle requête à partir de celle(s) en paramètre en traitant chaque dimen-

sion indépendamment les unes des autres. Nous présentons maintenant formellement ces opérateurs en commençant par les opérateurs binaires, qui sont les opérateurs ensemblistes classiques Garcia-Molina et al. (2008) étendus à plusieurs dimensions, ainsi qu'un opérateur de priorité.

Définition 3.1 *(Opérateurs binaires) Soient deux requêtes q_1 et q_2 et l'opérateur op $\in \{\cup_B, \cap_B, \setminus_B\}$, q_1 op $q_2 = \times_i(m_i(q_1)$ op $m_i(q_2))$. Soient deux requêtes q_1 et q_2, $q_1 \lhd q_2 = q_1$.*

Les opérateurs binaires sont au nombre de 3. $mostfreq$ retourne, pour une requête q donnée en paramètre, une requête pour laquelle seuls les membres les plus fréquents de q dans chaque dimension sont retenus. max retourne, pour une requête q donnée en paramètre, une requête pour laquelle seuls les membres les plus couvrants de q dans chaque dimension sont retenus, selon la hiérarchie de la dimension. lca retourne, pour une requête q donnée en paramètre, une requête pour laquelle seuls les plus petits ancêtres communs des membres de q dans chaque dimension sont retenus, selon la hiérarchie de la dimension.

Définition 3.2 *(Opérateurs binaires) Soit q une requête telle que $m_i(q) = \langle S_i, f_i\rangle$ pour tout i. $mostfreq(q) = \times_i\langle S'_i = \{m \in S_i | \nexists m' \in S_i, f_i(m') > f_i(m)\}, f_{i_{|_{S'_i}}}\rangle$ ($f_{i_{|X}}$ désigne la restriction de la fonction f_i à l'ensemble X). $max(q) = \times_i\langle S'_i = \{m \in m_i(q) | \nexists m' \in m_i(q), m <_i m'\}, f_{i_{|_{S'_i}}}\rangle$. Soit lca la fonction qui retourne, pour un ensemble donné de membres M de la dimension D_i, leur ancêtre commun selon $<_i$, i.e., $\{m \in D_i | \forall m' \in M, m' <_i m \wedge \nexists m'', m' <_i m'' m'' <_i m\}$. $lca(q) = \times_i lca(m_i(q))$.*

Exemple 3.2 *Considérons les deux premières requêtes de l'Exemple 3.1, nous avons :*

- $q_4 = q_1 \cup_B q_2 = \{Pepsi, Coke, Coke\} \times \{July08, July08\} \times \{Paris, Marseille, North, South\}$
- $q_5 = q_1 \cap_B q_2 = \{Coke\} \times \{July08\} \times \emptyset = \emptyset$
- $q_6 = q_1 \setminus_B q_2 = \{Pepsi\} \times \emptyset \times \{Paris, Marseille\} = \emptyset$
- $mostfreq(q_4) = \{Coke, Coke\} \times \{July08, July08\} \times \{Paris, Marseille, North, South\}$
- $max(q_4) = \{Pepsi, Coke, Coke\} \times \{July08, July08\} \times \{North, South\}$
- $lca(q_4) = \{Soda\} \times \{S2\text{-}08\} \times \{France\}$

Considérons le log L composé des trois requêtes $L = \langle q_1, q_2, q_3\rangle$. Si nous résumons ce log en ne gardant que les références communes à toutes les requêtes, nous obtenons : $q_s^1 = q_1 \cap_B q_2 \cap_B q_3 = \emptyset$. Si nous résumons en fonction des fréquences sur ces requêtes, nous obtenons : $q_s^2 = mostfreq(q_1 \cup_B q_2 \cup_B q_3) = \{Coke, Coke\} \times \{July08, July08\} \times \{North, North, South, South\}$. Si nous résumons en fonction du lca, nous obtenons : $q_s^3 = lca(q_1 \cup_B q_2 \cup_B q_3) = \{Soda\} \times \{2008\} \times \{France\}$.

3.3 Propriétés des opérateurs de QSL

Nous donnons maintenant quelques propriétés simples et aisément démontrables vérifiées par les opérateurs de QSL. Certaines de ces propriétés, comme la distributivité de max ou la commutativité de max et lca sont utilisées dans l'implémentation de notre cadre. Soient q, q_1, q_2 des requêtes. Les opérateurs $\cup_B, \cap_B, \setminus_B$ conservent leurs propriétés classiques. max et $mostfreq$ sont idempotents : $max(max(q)) = max(q)$ et $mostfreq(mostfreq(q)) = mostfreq(q)$. max est distributif sur $\cup_B$ et sur $\setminus_B$: $max(q_1 \cup_B q_2) = max(max(q_1) \cup_B max(q_2))$ et

$max(q_1 \setminus_B q_2) = max(q_1 \setminus_B max(q_2))$. $\lhd$ est associatif : $q \lhd (q_1 \lhd q_2) = (q \lhd q_1) \lhd q_2 = q$. max et lca commutent : $max(lca(q)) = lca(max(q)) = lca(q)$. Finalement, $mostfreq(lca(q)) = lca(q)$.

4 Mesure de qualité d'un résumé

Cette section présente la mesure utilisée pour évaluer la qualité d'un résumé. Nous introduisons cette mesure intuitivement puis donnons sa définition formelle, avant de conclure sur les propriétés des opérateurs de *QSL* vis-à-vis de cette mesure.

4.1 Intuition

Nous cherchons à évaluer si une requête (resp., un log), qui correspond à un ensemble de références (resp., requêtes), résume fidèlement une autre requête (resp., un autre log). Les opérateurs de *QSL* résument en conservant, ajoutant ou retirant des références aux requêtes à résumer pour constituer le résumé. Une manière d'évaluer cette fidélité est de mesurer la proportion de ce qui est ajouté ou enlevé par l'opérateur. Nous nous sommes donc tournés vers les mesures de rappel et précision classiques en recherche d'information. Dans notre cas, ces mesures devraient être étendues pour prendre en compte la relation de couverture exploitée par les opérateurs. Par exemple, considérons l'expression $lca(q_1 \cup q_2)$ de l'exemple 3.2 résumant q_1 et q_2 par $\{Soda\} \times \{S2\text{-}08\} \times \{France\}$. Cette expression résume en retirant toutes les références de q_1 et q_2 et ajoute la référence $r = \{Soda\} \times \{S2\text{-}08\} \times \{France\}$, ce qui donne un rappel et une précision nulle. Or on pourrait considérer que ce résumé est fidèle dans le sens où la référence ajoutée couvre les références retirées ; le rappel serait donc excellent et la précision dépendrait du nombre de références couvertes par la référence ajoutée r et non présentes dans les références retirées de q_1 ou q_2.

Ainsi, nous proposons d'étendre rappel et précision par la prise en compte d'une relation de couverture. Pour être le plus général possible, nous choisissons la relation de couverture définie sur les références puisque tant les requêtes que les logs peuvent être assimilés à des ensembles de références. Ainsi définie de manière générale, cette mesure présente l'intérêt de pouvoir être appliquée sur deux requêtes ou deux logs, ou n'importe quel couple d'ensembles de références.

4.2 Définitions et propriétés

Nous définissons tout d'abord la notion de couverture d'une référence par une autre et de couverture d'un ensemble de références.

Définition 4.1 *(Couverture) Une référence r couvre une référence r' si $r = \langle m_1, ..., m_n \rangle$, $r' = \langle m'_1, ..., m'_n \rangle$ et $\forall_i \in [1, n]$, $m_i >_i m'_i$ ou $m_i = m'_i$. Pour un ensemble de références R, $cover(R) = \{f \in \Pi_{L_1^1}(D_1) \times \Pi_{L_1^2}(D_2) \times ... \times \Pi_{L_1^n}(D_n) \mid \exists r \in R, r \; covers \; f\}$*

Reprenons l'exemple plus haut. $L = q_1 \cup q_2$, $R = lca(q_4)$. Appelons $K = L \cap R$, $D = L \setminus K$ et $A = R \setminus K$. $K = \emptyset$ et $cover(A) = \{Pepsi, Coke, Orangina\} \times \{July08, August08, September08, October08, November08, December08\} \times \{Paris, Blois, Marseille\}$ avec $|cover(A)| = 54$. $cover(D) = \{Pepsi, Coke\} \times \{July08\} \times \{Paris, Marseille\} \cup \{Coke\} \times$

$\{July08\} \times \{Blois\}$ et $|cover(D)| = 5$. Nous avons $cover(D) \subset cover(A)$. Dans ce cas, on s'attend à un rappel maximum et une précision mauvaise.

Ainsi, nous proposons un rappel mesurant la proportion de références de $cover(D)$ trouvées dans $cover(A)$ par rapport aux références de $cover(D)$, tout en favorisant un K maximal. Dualement, une précision mesurera la proportion de références de $cover(D)$ trouvées dans $cover(A)$ par rapport aux références de $cover(A)$, tout en favorisant un K maximal. Bien évidemment, si le résumé est l'ensemble vide, alors nous considérons que $p = r = 0$.

Définition 4.2 *(hf-measure) Soient L et R deux ensembles tels que $K = L \cap R$, $D = L \setminus K$ et $A = R \setminus K$ et $\{D_1, ..., D_n\}$ un ensemble de dimensions permettant de calculer la couverture. Le h-recall est défini par $r = \frac{|K \cup (cover(D) \cap cover(A))|}{|K \cup cover(D)|}$ et la h-precision est définie par $p = \frac{|K \cup (cover(D) \cap cover(A))|}{|K \cup cover(A)|}$. Ces deux mesures sont agrégées par une F-mesure classique, hf-measure $(L, R, \{D_1, ..., D_n\}) = 2 \times \frac{p \times r}{p + r}$.*

Notons que tous les opérateurs de *QSL* maximisent soit le h-recall ou soit la h-precision : l'opérateur $\cup_B$ et l'opérateur lca conduisent à un h-recall de 1, la h-precision étant entre 0 et 1, et tous les autres opérateurs conduisent à une h-precision de 1, le h-recall étant entre 0 et 1. Finalement, la propriété suivante est aisément démontrable.

Propriété 4.1 *(hf-measure maximale) Soient L et R deux ensembles et $\{D_1, ..., D_n\}$ un ensemble de dimensions permettant de calculer la couverture. hf-measure $(L, R, \{D_1, ..., D_n\}) = 1$ si et seulement si R et L couvrent exactement le même ensemble de références.*

5 Résumé d'un log

Dans cette section, nous présentons les algorithmes permettant de résumer un log en utilisant *QSL* et la mesure de qualité définie ci-dessus.

5.1 Algorithmes de résumé

L'algorithme SummarizeLog est un algorithme glouton qui résume un log en résumant successivement des requêtes au moyen des opérateurs de *QSL*, jusqu'à atteindre une taille donnée. L'expression *QSL* employée est celle qui maximise la mesure de qualité tout en faisant varier le log. Le choix de l'expression utilise deux stratégies. La première teste pour chaque requête ou couple de requêtes consécutives quelle opération de *QSL* maximise la qualité, l'expression *QSL* est alors l'application de cette opération (stratégie 1, présentée en annexe). La seconde teste pour chaque couple de requêtes consécutives quelle opération binaire de *QSL* maximise la qualité, applique cette opération, puis teste sur le résultat quel opérateur unaire maximise la qualité. L'expression *QSL* est alors de la forme $u(q\ b\ q')$ où u est un opérateur unaire et b un opérateur binaire (stratégie 2).

Un résumé est donc également un log. Dans ce qui suit, si $S = \langle q_1, \ldots, q_m \rangle$ est un résumé, nous appelons $query(q_i)$ l'ensemble des requêtes impliquées dans l'expression *QSL* de q_i. La qualité testée à chaque itération de $SummarizeLog$ est une qualité "locale" entre la ou les requêtes intervenant dans l'expression *QSL* et le résultat de l'expression. Une qualité "globale" pourra être évaluée entre le log initial et son résumé.

5.2 Propriétés des résumés

Notons tout d'abord qu'un résumé S d'un log L définit une partition de L. Par construction, chaque requête de S est définie par une expression *QSL* impliquant une sous-séquence distincte de requêtes de L.

Propriété 5.1 *(Partitionnement) Un résumé $S = \langle s_1, \ldots, s_m \rangle$ d'un log L définit une partition de L où chaque s_i résume par une expression* QSL *une sous-séquence non vide de L, les séquences ainsi résumées étant deux à deux disjointes et recouvrant l'intégralité de L.*

En utilisant les propriétés des opérateurs de *QSL*, nous identifions deux sous-langages particuliers appelés respectivement *QSL*^{r} et *QSL*^{p}. *QSL*^{r} est le langage composé des opérateurs maximisant le h-recall i.e., *QSL*^{r} =$\{\cup_B, lca\}$ et *QSL*^{p} est le langage composé des opérateurs maximisant la h-precision, i.e., *QSL*^{p} =$\{\cap_B, \setminus_B, \lhd, mostfreq, max\}$. Ces deux langages conduisent à deux propriétés simples. Appelons pour une requête q, $member(q)$ l'ensemble des membres apparaissant dans q, i.e., $member(q) = \bigcup_i m_i(q)$ et pour un ensemble de requêtes X, $member(X) = \bigcup_{q \in X} member(q)$.

Propriété 5.2 *(Requête obtenue avec* QSL^r*) Soit q^r une requête définie par une expression de* QSL^r *et m un membre. S'il n'existe pas de membre $m' \in member(q^r)$ tel que $m' \geq m$ alors $m \notin member(query(q^r))$ et $\nexists m'' \in member(query(q^r))$ tel que $m > m''$. Par contre si $\exists m' \in member(q^r)$ tel que $m > m'$ alors $\exists m'' \in member(query(q^r))$ tel que $m > m''$.*

Cette propriété indique notamment que si un résumé est construit uniquement à l'aide d'opérations maximisant le h-recall, tout membre non couvert par un membre du résumé n'apparaît pas dans les requêtes intervenant dans l'expression. Une propriété duale concerne la h-precision.

Propriété 5.3 *(Requête obtenue avec* QSL^p*) Soit q^p une requête définie par une expression de* QSL^p *et m un membre. Si $m \in member(q^p)$ alors $m \in member(query(q^p))$.*

Ces deux propriétés s'étendent trivialement aux résumés.

Propriété 5.4 *(Résumé obtenu avec* QSL^r*) Soit S^r un résumé construit avec des expressions* QSL^r *à partir d'un log L. Si un membre m n'est pas couvert par un membre de S^r, alors ni m ni aucun m' couvert par m n'apparaît dans L. Par contre si m couvre des membres de S^r, m couvre des membres de L.*

Propriété 5.5 *(Résumé obtenu avec* QSL^p*) Soit S^p un résumé construit avec des expressions* QSL^p *à partir d'un log L. Un membre m apparaissant dans S^p apparaît nécessairement dans L.*

La section suivante illustre l'intérêt de ces propriétés.

6 Interrogation d'un log

Dans cette section, nous illustrons l'utilisation des propriétés données ci-dessus pour interroger efficacement un log, en utilisant des résumés pour accéder uniquement à des sous-ensembles du log. Nous considérons des recherches simples sur un log, comme par exemple : existe t-il des

requêtes du log qui comportent le membre m, ou un membre couvert par m ? Quelles sont les requêtes du log qui comportent le membre m, le croisement $m \times m'$, ou un membre couvert par m ? Quelles sont les requêtes du log qui suivent une requête comportant le membre m ? Ces recherches sont exprimées à l'aide des opérations suivantes.

Définition 6.1 *(Recherche dans un log) Soit L un log et m un membre. lookup(m)* $\equiv \exists q \in queries(L), m \in members(q)$. *lookupDescendant(m)* $\equiv \exists q \in queries(L)$, $\exists m' \in members(q)$, $m > m'$. $getQueries(m) = \{q \in queries(L) | \exists m \in members(q)\}$. $getCoveredQueries(m) = \{q \in queries(L) | \exists m' \in members(q), m > m'\}$. $getSuccessors(m) = \{q \in queries(L) | \exists q' \in queries(L), m \in members(q'), \langle q', q\rangle \subseteq L\}$

Supposons que deux résumés soient disponibles pour un log L, le premier S^r construit avec *QSL*^{r} et le second S^p construit avec *QSL*^{p}. Nous présentons les deux algorithmes à la base de l'implémentation de ces opérations (donnés en annexe).

Considérons l'opération $lookup(m)$. Les propriétés 5.4 et 5.5 assurent que si m est présent dans S^p alors $lookup(m) = true$ et si m n'est pas couvert par un membre de S^r alors $lookup(m) = false$. Dans le cas contraire, les propriétés 5.1 et 5.2 assurent qu'il est suffisant de chercher parmi les requêtes de L figurant dans la définition des requête de S^r dont un membre couvre m. L'algorithme de $lookup$, basé sur ces propriétés, est donné ci-dessous. Il sert également de base à l'algorithme de $lookupDescendant$, où seules les lignes 1, 4 et 9 changent pour exploiter la dernière partie de la propriété 5.4.

Exemple 6.1 *Soit le log de l'exemple 3.1 et ses résumés $S^r = \langle q'_1, q'_2\rangle$ et $S^p = \langle q'_3\rangle$, où $q'_1 = lca(q_1) = \{Soda\} \times \{S2\text{-}08\} \times \{North, South\}$, $q'_2 = q_2 \cup_B q_3 = \{Coke, Orangina\} \times \{July08, S2\text{-}08\} \times \{North, South\}$, et $q'_3 = q_1 \lhd q_2 \lhd q_3 = q_1 = \{Pepsi, Coke\} \times \{July08\} \times \{Paris, Marseille\}$. L'appel à $lookup(Pepsi)$ uniquement à S^p pour répondre $true$ et l'appel à $lookup(France)$ nécessite uniquement d'accéder à S^p et S^r pour répondre $false$. L'appel à $lookup(Orangina)$ nécessite uniquement d'accéder à S^p et S^r pour répondre $true$ (cf. lignes 4 à 6). Pour répondre $false$, l'appel à $lookup(August08)$ nécessite d'accéder à S^p, S^r et finalement q_1, mais évite d'accéder à q_2 et q_3 puisque $August08$ ne peut pas apparaître dans les opérandes d'une union dont le résultat ne le contient pas (cf. lignes 3 à 6 de $candidateQueries$).*

L'algorithme de $getQueries$ se déduit aisément de celui de $lookup$, en supprimant les 6 premières lignes et en retournant les requêtes pertinentes de $candidateQueries$ au lieu d'un booléen. $getQueries$ permet également de trouver les requêtes où un croisement de membres $m \times m'$ apparaît, qui correspondent à $getQueries(m) \cap getQueries(m')$, et par extension les requêtes où des références apparaissent. $getQueries$ est à la base de $getCoveredQueries$ puisqu'il suffira d'ajouter pour cet opérateur le cas correspondant à la dernière partie de la propriété 5.4. Finalement, $getQueries$ sera utilisée pour l'opération $getSuccessors$ si les requêtes du log sont stockées dans l'ordre de leur apparition dans le log.

7 Implémentation et tests

QSL, hf-measure, SummarizeLog et lookup ont été implémentés en Java 6 pour résumer des logs de requêtes MDX. L'implémentation a été faite en considérant que les dimensions concernées résident en mémoire. Celles-ci sont représentées par des arbres stockant pour chaque

membre la cardinalité de sa couverture au niveau le plus fin. Les algorithmes de calcul de hf-measure se trouvent dans XXX (XXXa). Les tests ont été réalisés sur un Core 2 Duo E8400 à 3,00 GHz, avec 3,48 Go de RAM exploitables sous windows XP pro SP3. Les logs utilisés proviennent de logs synthétiques sur la base de données exemple FoodMart fournie avec le moteur OLAP Mondrian. La génération des logs, simulant une analyse OLAP, est expliquée dans neg. Les logs générés ont une densité assez élevée (5 dimensions parmi les 13 possibles sont utilisées pour naviguer). Ils sont répartis en 4 fichiers de respectivement 119, 241, 437 et 904 requêtes. Par manque de place, seuls quelques tests sont présentés. Dans ce qui suit, les tailles de résumés sont exprimées en fraction de la taille du log original.

Le premier test indique la fréquence d'apparition sur une échelle logarithmique, de chaque opérateur de *QSL* dans les expressions construites par SummarizeLog. La figure 2 montre que pour *QSL*, la mesure hf-measure tend à privilégier l'opération d'union et le lca. Pour QSL^p, mostfreq n'est pas utilisé car les opérateurs ensemblistes de QSL^p n'engendrent pas de multi-ensembles. De manière générale, aucun opérateur n'est jamais utilisé.

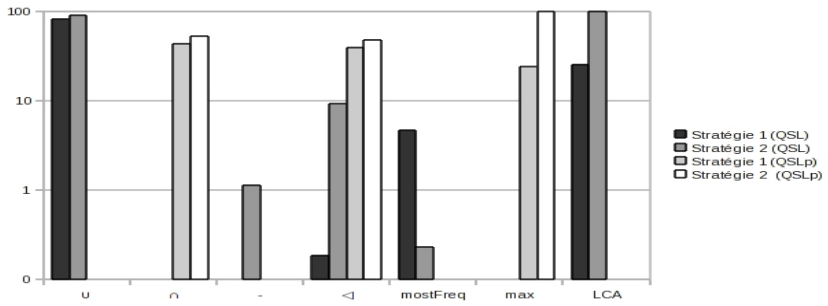
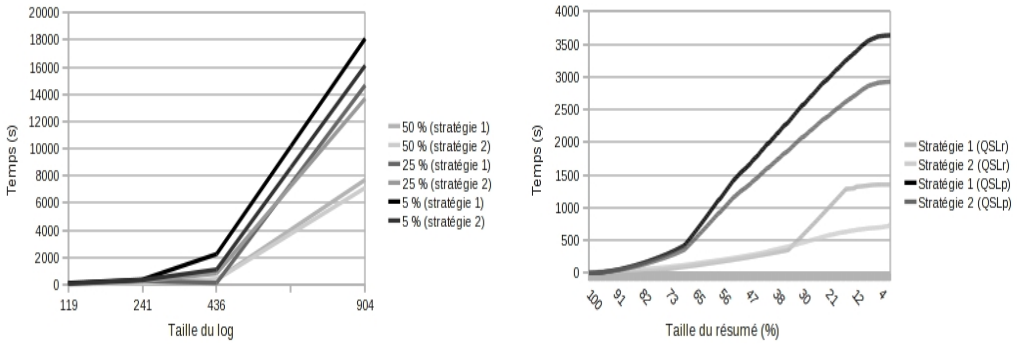


FIG. 2 – *Fréquence d'apparition des opérateurs de* QSL *et* QSL^p

Le test suivant concerne l'efficacité de l'algorithme SummarizeLog dans ses deux stratégies présentées section 5. La figure 3 (a) montre que le temps mis par SummarizeLog évolue de manière polynomiale avec la taille du log à résumer et s'avère assez couteux pour des logs importants. Ce temps reste néanmoins bien plus acceptable pour QSL^r (figure 3 (b)) puisqu'il suffit de 10 minutes environ avec la deuxième stratégie pour réduire le plus gros fichier log de 80 %. De manière générale, la stratégie 2 reste plus efficace, demandant moins de tests de qualité (partie la plus couteuse de l'algorithme SummarizeLog).

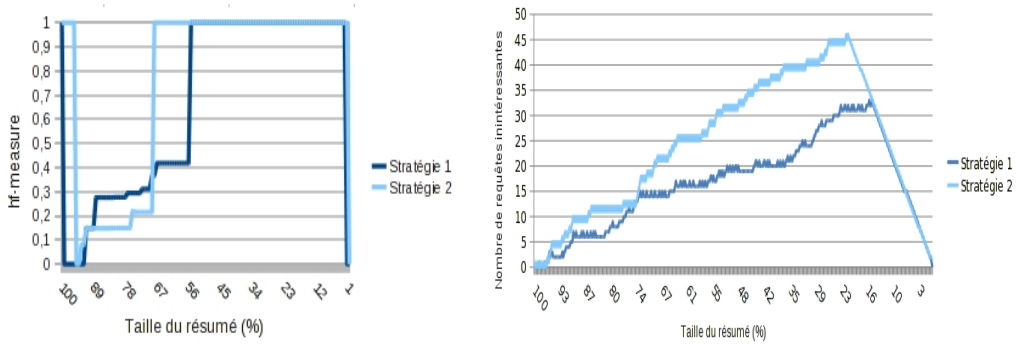
Nous nous focalisons maintenant sur QSL^r, qui est le langage permettant de générer les résumés à la base de la recherche dans le log. Le test suivant concerne la qualité globale des résumés obtenus avec QSL^r, d'une part en utilisant hf-measure (en l'occurrence h-precision) entre toutes les références d'un log initial et celles des résumés obtenus (conduit sur le log de 119 requêtes, figure 4 (a)), et d'autre part en regardant la proportion de requêtes inintéressantes dans les résumés obtenus, une requête inintéressante ayant uniquement le membre All dans toutes les dimensions (conduit sur le log de 437 requêtes, figure 4 (b)). La qualité globale se dégrade très vite (le rapport entre l'intersection des couvertures et la couverture donne une hf-measure très faible) ce qui est du à l'opération de lca, puis augmente grâce à l'opération d'union jusqu'à atteindre le maximum et s'y stabiliser. La dernière chute s'expliquant par l'utilisation d'un lca détériorant à nouveau la h-precision. Le nombre de requêtes inintéressantes augmente tout d'abord, parce qu'elles bénéfi-



(a) (b)

FIG. 3 – *Efficacité de SummarizeLog*

cient d'un bon h-recall, jusqu'à ce que leur présence dégrade trop la h-precision, elles sont alors « absorbées » par les autres requêtes. On constate finalement que la stratégie 1, qui utilise le lca moins systématiquement, donne de meilleurs résultats. Des tests menés avec *QSL* montrent un comportement similaire.



(a) (b)

FIG. 4 – *Qualité des résumés obtenus avec* QSL^r

Finalement, le dernier test concerne l'efficacité de lookup. La figure 5 montre le gain moyen en rapidité pour une recherche par lookup (3210 membres tirés aléatoirement) à partir des deux résumés calculés avec QSL^p et QSL^r de deux logs. Le gain est significatif pour de petits résumés. Notons toutefois que log comme résumés résident en mémoire.

8 Travaux existants et perspectives

Le résumé de données structurées est un sujet abordé dans de nombreux domaines comme les logs de serveurs web Zadrozny et Kacprzyk (2007), l'extraction de motifs (voir par exemple Ndiaye et al. (2010) qui inclut un état de l'art), les séquences d'évènements Peng et al. (2007), les bases de données Saint-Paul et al. (2005), les flux de données multidimensionnelles Pitarch et al.

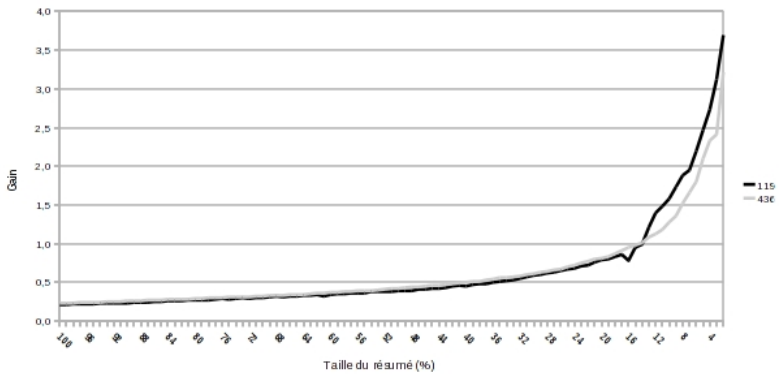


FIG. 5 – *Efficacité de lookup*

(2010) et les cubes de données Lakshmanan et al. (2002). Plusieurs de ces travaux sont basés sur la théorie des ensembles flous (Zadrozny et Kacprzyk (2007); Saint-Paul et al. (2005)) et/ou sont des techniques de compression pour lesquelles il est important que les données originales puissent être régénérées (Lakshmanan et al. (2002); Ndiaye et al. (2010)). En outre, il peut arriver que le résumé ne soit pas du même type que les données qu'il résume. Dans le domaine des bases de données (Saint-Paul et al. (2005); Lakshmanan et al. (2002)), le résumé est appliqué à l'instance de la base de données où, dans le cas de données OLAP, les valeurs des mesures sont prises en compte.

Dans cet article, nous proposons de résumer un log de requêtes OLAP, c'est à dire une séquence de requêtes lancées sur une base de données et non une instance de la base de données. Nous n'utilisons pas la théorie des ensembles flous pour résumer. Au lieu de cela, nous nous appuyons sur les hiérarchies décrites dans les tables de dimension. Le type du résumé est le même que celui des données résumées : une requête résume une autre requête et un log résume un autre log. Nous ne nous intéressons pas à la régénération des données à partir du résumé, mais à son utilisation pour interroger efficacement un log. Nos contributions s'appuient sur un langage de résumé de requêtes qui permet de spécifier un résumé, une mesure pour évaluer la qualité d'un résumé et un algorithme pour calculer automatiquement un résumé de log de requêtes de bonne qualité. Notre approche est implémentée pour résumer des logs de requêtes MDX et quelques tests préliminaires illustrent son intérêt.

A notre connaissance, il n'existe aucun travail traitant du problème de la représentation concise et exploitable d'un log de requêtes sur une base de données. Comme indiqué dans Khoussainova et al. (2009), de nombreux SGBD fournissent un service de logging de requêtes, principalement à des fins de tuning physique, et permettent à l'utilisateur de visualiser le log. Cependant, la manière dont le log est affiché, souvent sous forme de table ou de fichier, n'est pas adaptée à la navigation. Dans nos travaux précédents XXX (XXXb), nous proposons d'organiser un log de requêtes OLAP sous la forme d'un site Web. Mais, si le log est volumineux, naviguer dans ce site Web peut être fastidieux. Un outil efficace de visualisation et de navigation de log doit encore être développé et ce travail est un pas dans cette direction.

Nos futurs travaux devront étudier l'efficacité du cadre pour l'analyse en ligne. Nous aborderons notamment son extension à un contexte collaboratif où un log n'est pas seulement une

séquence de requêtes mais est composé de plusieurs séquences de requêtes lancées par différents utilisateurs.

Références

Ph. D. thesis.

Chatzopoulou, G., M. Eirinaki, et N. Polyzotis (2009). Query recommendations for interactive database exploration. In M. Winslett (Ed.), *SSDBM*, Volume 5566 of *Lecture Notes in Computer Science*, pp. 3–18. Springer.

Garcia-Molina, H., J. D. Ullman, et J. Widom (2008). *Database Systems : The Complete Book*. Upper Saddle River, NJ, USA : Prentice Hall Press.

Giacometti, A., P. Marcel, et E. Negre (2009a). Recommending multidimensional queries. In T. B. Pedersen, M. K. Mohania, et A. M. Tjoa (Eds.), *DaWaK*, Volume 5691 of *Lecture Notes in Computer Science*. Springer.

Giacometti, A., P. Marcel, E. Negre, et A. Soulet (2009b). Query recommendations for olap discovery driven analysis. In *DOLAP*, pp. 81–88.

Khoussainova, N., M. Balazinska, W. Gatterbauer, Y. Kwon, et D. Suciu (2009). A case for a collaborative query management system. In *CIDR*. www.crdrdb.org.

Lakshmanan, L. V. S., J. Pei, et J. Han (2002). Quotient cube : How to summarize the semantics of a data cube. In *VLDB*, pp. 778–789. Morgan Kaufmann.

Ndiaye, M., C. T.Diop, A. Giacometti, P. Marcel, et A. Soulet (2010). Cube based summaries of large association rule sets. In *sixth International Conference on Advanced Data Mining and Applications (ADMA)*. LNCS.

Peng, W., C. Perng, T. Li, et H. Wang (2007). Event summarization for system management. In P. Berkhin, R. Caruana, et X. Wu (Eds.), *KDD*, pp. 1028–1032. ACM.

Pitarch, Y., A. Laurent, et P. Poncelet (2010). Summarizing multidimensional data streams : A hierarchy-graph-based approach. In M. J. Zaki, J. X. Yu, B. Ravindran, et V. Pudi (Eds.), *PAKDD (2)*, Volume 6119 of *Lecture Notes in Computer Science*, pp. 335–342. Springer.

Saint-Paul, R., G. Raschia, et N. Mouaddib (2005). General purpose database summarization. In K. Böhm, C. S. Jensen, L. M. Haas, M. L. Kersten, P.-Å. Larson, et B. C. Ooi (Eds.), *VLDB*, pp. 733–744. ACM.

Stefanidis, K., M. Drosou, et E. Pitoura (2009). "You May Also Like" Results in Relational Databases. In *PersDB*.

XXX (XXXa). Xxx. In *XXX*.

XXX (XXXb). Xxx. In *XXX*.

Zadrozny, S. et J. Kacprzyk (2007). Summarizing the contents of web server logs : A fuzzy linguistic approach. In *FUZZ-IEEE*, pp. 1–6. IEEE.

Annexe : les algorithmes SummarizeLog et lookup

Algorithm 1 SummarizeLog (stratégie 1)

INPUT :
L : a log,
$\mathcal{U}$: a set of unary operators,
$\mathcal{B}$: a set of binary operators,
D : a set of dimensions,
α : a positive integer
OUTPUT : A summary of L.
VARIABLES : A query q
while $|L| > \alpha$) **do**
  $q = argmax_1(\{\text{hf-measure } (q', q'', D) | q'' = op(q'), op \in \mathcal{U}, q' \in L\} \cup \{\{\text{hf-measure } (q' \cup q'', q''', D) | q''' = op(q', q''), op \in \mathcal{B}, q', q'' \in L\})$
  replace in L $queries(q)$ by q
end while
return L

Algorithm 2 lookup

INPUT :
L : a log,
S^r : a summary of L constructed with QSL^r,
S^p : a summary of L constructed with QSL^p,
m : a member.
OUTPUT : A boolean.
if $m \in member(S^p)$ **then**
  return True
end if
if $\exists q \in query(S^r)$ with $q = q_1 \cup_B \ldots \cup_B q_x$ and $m \in member(q)$ **then**
  return True
end if
for each $q \in query(S^r)$ such that $\exists m' \in member(q)$ with $m' \geq m$ **do**
  for each $q' \in candidateQueries(q, m)$ **do**
    if $m \in q'$ **then** {Access to L}
      return True
    end if
  end for
end for
return False

Algorithm 3 candidateQueries

INPUT :
q : a query,
m : a member.
OUTPUT : A set of queries where m may appear.
VARIABLE : A set of queries Q, a set of members M.
$Q \leftarrow \emptyset$
let $lca(e_1) \cup_B \ldots \cup_B lca(e_x) \cup_B q_1 \cup_B \ldots \cup_B q_y$ be the *QSL* expression defining q
$M \leftarrow \{m' \in member(q) | m' \geq m\}$
if $m \in M$ **then**
  $Q \leftarrow Q \cup \{q_1, \ldots, q_y\}$
end if
for each $m' \in M$ **do**
  for each q' appearing in $lca(e_1) \cup_B \ldots \cup_B lca(e_x)$ **do**
    if q' appears in a number of compositions of lca $\leq level(m') - level(m)$ **then**
      $Q \leftarrow Q \cup \{q'\}$
    end if
  end for
end for
return Q

Summary

Leveraging query logs benefits the users analyzing large data warehouses with OLAP queries. But so far nothing exists to allow the user to have concise and usable representations of what is in the log. In this paper, we propose a method for summarizing OLAP query logs. The basic idea is that a query summarizes another query and that a log, which is a sequence of queries, summarizes another log Our framework includes a simple algebra to declaratively specify a summary, and a measure to assess the quality of the summaries. We also propose several strategies for automatically computing a good quality summary of a query log, and we show how some simple properties on the Tests on MDX query logs showed the usefulness of our approach.