PerCubes : Une méthodologie efficace pour la recherche de cubes pertinents dans les systèmes OLAP

Rahma Djiroun*, Kamel Boukhalfa*, Sandro Bimonte**, Zaia Alimazighi*

*LSI, USTHB,BP 32 EL ALIA 16111 BAB EZZOUAR
ALGER. (rdjiroun,kboukhalfa,zalimazighi)@usthb.dz
**IRSTEA, UR TSCF 1s24 Avenue des Landais, 63172,
France sandro.bimonte@irstea

Résumé. Aujourd'hui les Entrepôts de données (ED) et les systèmes OLAP (On Line Analytical Processing) représentent une solution incontournable pour l'analyse décisionnelle des gros volumes de données pour les entreprises et administrations. Souvent, au niveau d'une entreprise/organisation se trouvent plusieurs axes d'activités qui nécessitent plusieurs cubes de données différents. La recherche d'informations transversales sur ces cubes de données n'est pas possible aujourd'hui car aucune méthodologie pour la recherche de cubes pertinents n'a été proposée dans la littérature scientifique. Dans ce papier, nous proposons un cadre conceptuel pour rechercher des cubes de données pertinents. Une implémentation dans une architecture ROLAP est aussi proposée pour valider notre approche théorique.

Mots clés : OLAP, Cube de données, recherche de cubes, classes de pertinences.

1 Introduction

Les Entrepôts de Données (EDs) constituent une solution efficace pour l'analyse multidimensionnelle des grands volumes de données en ligne. Nous assistons actuellement à une prolifération des cubes et entrepôts de données en ligne[1 2]. Les modèles multidimensionnels permettent la structuration de ces données en définissant des dimensions (telles que *le temps*, *les produits*, etc.) et des faits (les sujets d'analyse) (par exemple *la production laitière*). Cette structuration a pour objectif d'observer les faits à travers des mesures (par exemple, *quantité de lait produite*), en fonction des dimensions qui représentent les axes d'analyse. Les systèmes OLAP permettent d'explorer les cubes de données, à l'aide d'un ensemble d'opérateurs, pour découvrir des résultats ou des tendances inattendues, ou encore pour valider des hypothèses décisionnelles.
Habituellement, au niveau d'une entreprise ou d'une organisation se trouvent plusieurs activités qui nécessitent l'exploitation de différents cubes de données. Le processus

[1] http ://statistics.amis-outlook.org/data/index.html

[2] https ://visionet.franceagrimer.fr / Pages / SourcesEtMethodes.aspx ? sousmenu = contexte%20g %C3 %A9n%C3 %A9ral

décisionnel associé à une activité est donc fortement lié à un cube de données spécifique. En même temps, la maturité des systèmes OLAP a conduit à l'adoption de ces technologies d'aide à la décision dans la plupart des moyennes et grandes entreprises avec un nombre toujours plus grandissant de cubes de données par secteur d'activité ou par projet. Si le croisement des données au sein d'un seul cube de données représente une véritable source d'information, les bénéfices décisionnels associés à une analyse transversale des différents cubes de données restent inexploités. En effet, les décideurs peuvent faire émerger des besoins décisionnels occasionnels qui ne reposent pas nécessairement sur un seul cube, mais qu'ils portent sur un ou plusieurs cubes de données (Cabibbo et Torlone, 2004) en utilisant l'opérateur OLAP *Drill-Accros*. Par contre, aucune méthodologie pour la recherche des cubes de données les plus pertinents n'a été proposée. Aussi, la consultation manuelle des métadonnées associées à chaque cube de données peut se révéler longue, complexe et souvent inutilisable vu la spécificité du domaine d'application. Cette problématique est encore plus importante vu la diffusion des cubes de données sur le web grâce à l'open-data[3] (Berro et al., 2013).

Si différents auteurs s'intéressent à la proposition des requêtes OLAP, ils ne l'abordent qu'au sein du même cube de données (Giacometti et al., 2008; Jerbi et al., 2011). Dans ce papier, nous proposons un cadre conceptuel pour la recherche de cubes de données à partir d'une requête exprimée par un ensemble de termes décisionnels en langage naturel. Il s'agit donc d'une approche qui permet l'extraction d'un ensemble de cubes pertinents à partir d'une collection de cubes de données. Une implémentation dans une architecture OLAP relationnel est aussi proposée pour valider notre approche théorique.

Le papier est organisé comme suit : dans la section 2, nous étudions les travaux connexes. Nous présentons un cas d'étude pour illustrer notre approche et nous nous concentrons sur le besoin des utilisateurs de cubes de données dans la section 3. Dans la section 4, nous présentons quelques définitions et terminologies. Nous présentons en détail notre approche de recherche de cubes dans la section 5, ainsi que l'ordre de pertinence que nous proposons. La section 6 présente notre outil pour la recherche de cubes de données ainsi quelques tests d'évaluations. Nous concluons dans la section 6 en présentant quelques perspectives.

2 Etat de l'art

Avec l'énorme croissance des besoins d'analyse des entreprises, OLAP utilise des vues multidimensionnelles des données agrégées pour fournir un accès rapide à l'information stratégique pour la prise de décision. Les vues multidimensionnelles sont également connues comme cube de données. Avec le grand nombre de cubes conçus dans les entreprises, l'analyse est devenue une tâche de plus en plus difficile. Dans cette section, nous présentons les travaux qui portent sur l'interrogation des cubes de données via le langage naturel et la recommandation de requêtes OLAP. (Feki et al.,

[3] https://www.data.gouv.fr/fr

2006) proposent un outil basé sur des patrons multidimensionnels en langage naturel. L'outil proposé permet de générer des requêtes exprimées sur les dimensions (axes d'analyses) et les soumettent aux décideurs pour sélectionner les axes d'analyses significatifs pour ces besoins. Ensuite, pour chaque axe d'analyse retenu, il génère un ensemble de requêtes exprimées sur les hiérarchies. Ces dernières sont présentées aux décideurs pour choisir celles qui répondent à leurs besoins. Un traitement sémantique de la requête a été abordé dans les travaux de (Kuchmann-Beauger et al. , 2011). Ce travail est basé sur le traitement du langage naturel pour les entrepôts de données. Il propose une interface en langage naturel dans laquelle l'utilisateur écrit une requête, qui sera enrichie en utilisant un thésaurus, puis reformulée si le système ne trouve pas de réponse.

Généralement les utilisateurs des systèmes OLAP naviguent dans un cube en lançant une séquence de requêtes sur un entrepôt de données. Ces requêtes sont stockées dans un fichier log dans le serveur. (Giacometti et al., 2008) ont utilisé ce fichier pour exploiter l'ensemble des séquences de requêtes précédentes et la séquence de requêtes courante. Les auteurs ont proposé un cadre générique pour recommander des requêtes OLAP à l'utilisateur et en particulier des requêtes MDX. (Negre, 2009) a proposé une instanciation du cadre générique proposé par (Giacometti et al., 2008). L'auteur, propose un prototype pour la recommandation et la classification des requêtes. Les requêtes de chaque session issues du fichier log sont regroupées en classes en se basant sur une distance calculée entre requêtes. La session exprimée en termes de classes de requêtes est appelée session généralisée. La classification se base sur la théorie des graphes traitant le problème de calcul du plus court chemin et utilise l'algorithme de clustering (K-mediods). Pour ordonner les requêtes candidates à la recommandation, l'auteur propose un algorithme qui s'appuie sur le profil d'utilisateur.

Afin de renvoyer à l'utilisateur un résultat en fonction de ses préférences, dans le cadre de la personnalisation dans les systèmes OLAP, (Jerbi et al., 2011) ont proposé une modélisation au sein d'une constellation prenant en compte l'expertise et les préférences de l'utilisateur. L'expertise du décideur est modélisée sous la forme d'annotations (commentaires, discussions, prises de décision, etc.). Les préférences sont exploitées pour des recommandations contextuelles annotées qui assistent l'usager dans son exploration de l'espace multidimensionnel. Dans le même contexte, (Golfarelli et al., 2011) proposent une approche où les préférences sont utilisées pour annoter la requête. Ils ont défini une algèbre qui permet la formulation de préférences sur les attributs, les mesures et les hiérarchies. La technique utilisée consiste à personnaliser une requête en traitant avec une sous-requête.

(Khemiri, R. et al. , 2013) proposent un système de recommandation de requêtes qui permet d'assister les utilisateurs pendant l'écriture de leurs requêtes. Le système est basé sur un ensemble de mesures sélectionnées par l'utilisateur et de requêtes du fichier log. Il permet de trouver les associations entre les requêtes du fichier log en utilisant la méthode des règles d'association pour extraire l'ensemble des items fréquents des attributs de dimensions selon l'ensemble de mesures sélectionnés par l'utilisateur.

(Kozmina, N, 2013) propose une approche pour la recommandation des rapports. Elle a développé un composant de recommandation de l'outil de reporting OLAP. L'approche proposée est composée de deux méthodes, la méthode de démarrage à froid qui est appliqué si un utilisateur soit nouveau dans le système ou classées comme passif, et la méthode de démarrage à chaud qui est appliquée pour les utilisateurs actifs du système.
(Umagandhi, R. et al., 2014) proposent une approche basée sur les méta-heuristiques pour recommander des requêtes en utilisant le moteur de recherche de requêtes journal. L'approche recommande des requêtes pour plusieurs types d'utilisateurs. Il permet de générer les modèles de requêtes fréquemment produites, à travers lequel l'utilisateur reçoit les requêtes recommandées. Les requêtes recommandées sont classées sur la base des préférences et l'heure à laquelle sont faites.

Table 1. Survey sur les travaux connexes – Une Comparaison.

Travaux de Recherche			Giacometti et al., 2008	Jerbi et al., 2011	Golfarelli et al., 2011	Fekki et al., 2012	Kuchmann et al., 2012	Khemiri et al., 2013	Kozmina, 2013	Umagandhi et al., 2014	Notre Approche
Entrée	Type de Requête	MDX	X	X	X	X				X	
		Langage naturel				X	X				X
		SQL							X		
	Fichier Log		X	X	X			X		X	
	Profile utilisateur			X	X			X	X	X	
	Collection de cubes										X
Sortie	Requêtes		X	X	X	X	X	X	X	X	
	Agrégation de mesures			X						X	
	cube/(s)										X
	Liste Ordonnée				X				X	X	X
Objectif	Performance			X				X		X	X
	Pertinence			X				X		X	X

Nous présentons un tableau comparatif (tableau 1) confrontant la panoplie des approches proposées. Nous définissons certains critères que nous jugeons pertinents pour étudier les travaux de recherches similaires de notre approche : les données en entrée, les données en sortie, l'ordonnancement et les objectifs.
Le critère de données en d'entrée consiste à définir les utilisés pour effectuer la recherche. Les données en entrée du processus de recommandation de requêtes peuvent être le profil utilisateur, le fichier Log, type de requêtes, etc. Par exemple, les utilisateurs peuvent exprimer leurs besoins par une requête. Cette requête peut être écrite en MDX, langage naturel ou par un ensemble de termes, etc.
Pour le critère de données en sortie l'utilisateur peut avoir des besoins en termes de requêtes, de mesures ou de cubes. Le critère objectif présente l'objectif de chaque approche en termes de performance ou de pertinence.

D'après le tableau, nous constatons qu'aucun de ces travaux n'aborde l'ordre de pertinence des cubes et la recherche de cubes dans une collection de cubes de données. Nous proposons, dans ce travail, une approche pour résoudre ce problème. Notre approche permet de retourner un ensemble de cubes ordonnés selon leur pertinence par rapport au besoin exprimé par l'utilisateur.

3 Cas d'étude

Dans cette section, nous présentons un cas d'étude qui sera utilisé dans le reste de cet article pour illustrer notre approche. Il est adapté à partir du projet national EDEN, qui concerne l'analyse des consommations énergétiques des exploitations agricoles (Bimonte et al., 2012). Des mesures de consommation et un relevé des pratiques est effectué sur une exploitation agricole pilote, dont les activités sont diverses : production laitière (*chèvres*, *vaches*), production agricole (*céréales*, *tomates*, *avoine*, *bersim*, etc.), production animal (*viande*, *poulet*, *œuf*, …etc.) et ainsi que la vente de ces produits sur le territoire national. Vu la diversité des activités au niveau de cette exploitation pilote, un grand volume de données est accumulé au cours du temps. Pour permettre l'analyse multidimensionnelle de ces activités, plusieurs cubes de données ont été conçus. Nous présentons dans ce papier trois cubes de données afin d'illustrer notre approche : le cube *production laitière* qui permet d'analyser l'activité liée à la production laitière ; le cube *production agricole* pour l'analyse des productions et l'usage des pesticides ; et le cube *vente* pour l'analyse de la vente des produits. Ces cubes sont destinés à trois types de décideurs : les responsables de l'atelier de production laitière, les experts d'ACV (Analyse Cycle de Vie) pour comprendre l'impact de l'utilisation des pesticides sur les cultures, et les gestionnaires économiques pour la définition des politiques de marketing. Les cubes ont des dimensions et des mesures différentes, mais aussi des dimensions en commun telles que *temps*, *production*, etc. Les détails de ces cubes sont présentés dans le Tableau 2.

Tableau 2. Cubes issus de cas d'étude d'une exploitation agricole pilote

Nom de cube	Description	Fait	Mesures [Agrégats]	Dimensions
Cube production laitière (C1)	analyser la quantité de lait produite.	Production Laitière	quantité de lait produite [total, moyenne], nombre animaux [total] quantité intrant [total], quantité extrant [total, moyenne].	opérateurs, temps [jour, mois, années], Elevages (vache, chèvre), production, produits, équipements.
Cube Production Agricole (C2)	analyser la production et l'usage des pesticides.	Production Agricole	Quantité intrant [total, moyenne], quantité extrant [total, moyenne], concentration pesticide [pourcentage%], flux pesticide [taux].	opérations techniques, cultures, temps [heure, jour, mois, compagne agricole], parcelles, produits, équipements, pesticides, production.

Cube Vente (C3)	Analyser les ventes.	Ventes	quantité vendue [total, max, moyenne], quantité commandée [max, moyenne, total].	client [type], temps [semaine, mois, années], production [catégorie], produits, localisation [ville, département, région], commandes.

Supposons, qu'un décideur soit intéressé par l'analyse des cubes qui concernent *les pesticides par production (Q1)* et par *le total de la quantité de lait produite par élevage (Q2)*. Il sera donc obligé de chercher, parmi toutes les documentations des cubes de données déployés pour l'exploitation agricole, les cubes les plus adaptés à son analyse. Ce qui correspond donc à examiner toutes les mesures et les niveaux de dimensions disponibles, avant de pouvoir conduire ces analyses. Cela implique une importante dépense en termes de temps, efforts et évidement importante perte économique. Dans la suite du papier nous utiliserons les requêtes *Q1* et *Q2* pour illustrer notre approche.

4 Concepts et Terminologie

Dans cette section, nous présentons quelques définitions préliminaires nécessaires pour comprendre notre approche.

Définition 1. (Collection): La collection est un ensemble de cubes déployés. Nous définissons C une collection de n cubes C_1, C_2, ..., C_n.

Définition 2. (*Composant Structurel* (CS)): les composants structurels représentent les termes qui décrivent les éléments conceptuels d'un cube. Nous appelons composant structurel : un Fait (F), une mesure [agrégat] (M [A]), Dimension (D) ou un niveau (N)

Définition 3. (*Cube):* Un cube est un ensemble de données construites à partir d'une partie d'un entrepôt de données, organisée dans une structure multidimensionnelle définie par un ensemble de composants structurels.

Un cube Cj est représenté par un ensemble de composants structurels : un Fait (F), un ensemble de mesures [agrégat] (M [A]), un ensemble de dimensions (D) organisée sur une hiérarchie de niveaux (N), tel que:$Cj=< F_j, M_j[A]^+, D_j[L_j]^+>$

Exemple 1. Le cube C_2 de tableau 1 est composé de Fait : *Production Laitière*, Mesures : *quantité de lait produite [total, moyenne], nombre animaux [total], quantité intrant [total]* and *quantité extrant [total, moyenne]* et Dimensions: *opérateurs, temps [jour, mois, années], Elevages (vache, chèvre), production, produits* and *équipements.* (voir tableau 1).

Soit le cube Ci, nous définissons $CS_D(C_j)$, $CS_F(C_j)$, $CS_N(C_j)$, $CS_M(C_j)$, $CS_A(C_j)$ l'ensemble de toutes les instances de CS "*Dimension*", "*Fait*", "*Niveau*", "*Mesure*" et "*Agrégat*" dans C_j respectivement. L'ensemble de tous les CS d'un cube Cj est définie comme suit: $CS(C_j)= CS_D(C_j) \cup CS_F(C_j) \cup CS_N(C_j) \cup CS_M(C_j) \cup CS_A(C_j)$.

Exemple 2. $CS_D(C_2)$ = {*opérateurs, temps, Elevages, production, produits, équipements*}

Définition 4. (*Catalogue):* Nous définissons le catalogue, l'ensemble des instances de tous les CS dans l'ensemble de cubes Cj. Soit C={C_1…C_n} l'ensemble des cubes disponibles, le catalogue CS (C) est défini comme suit: CS(C)=CS(C_1) ∪… ∪ CS(C_n).

Définition 5. (Requête utilisateur (Q)): La requête de l'utilisateur est exprimée par un ensemble de termes (t1, ..., tn) en langage naturel séparés par des opérateurs logiques tels que :Q(t_1 **OP_1** t_2 **OP_2** t_3**OP_3**… **OP_{n-1}** t_n) où $OP_i \in$ {AND, OR}.

Exemple 3. Q1(*pesticides* **and** *productio).*

Définition 6. *Similarité :* la similitude est une fonction qui permet de quantifier la similarité entre deux termes. Deux types de similitude sont définis : la similarité lexicale et similarité sémantique. Dans ce papier, nous considérons que la *similarité lexicale*, les termes sont similaires lexicalement si elles ont une séquence de caractères similaires (matching string). Plusieurs mesures ou distances de similarités sont mis en œuvre pour déterminer le taux de similarité entre deux termes (Wael H. Gomaa et al.2013) . Par example: Damerau-Levenshtein, Jaro, Jaro-Winkler, Needleman-Wunsch, Jaccard similarity..etc.

Exemple 4. Si l'on considère la requête précédente **Q_1**(*pesticides* **and** *production*)

Utilisant la distance de Jaro-Winkler, les instances de composants structurels qui sont similaires lexicalement depuis le catalogue de terme « *pesticides* » sont : (“*Pesticide”*, “*Pesticide flow”*, “ *Pesticide concentration”*, *...etc.*).
Pour le terme “*productio*”, les instances de composants structurels qui sont similaires lexicalement depuis le catalogue sont : (“*Production”*, *“Produit ”*, “ *Production laitière”*, *...etc.*).
Définition 7. *Requête Reformulée (Q'):* Requête reformulée est une requête, dont chaque terme *ti* de cette requête est remplacé par un terme similaire *ti*'. Le terme *ti'* est une instance d'un composant structurel du catalogue similaire au terme *ti* et choisi par l'utilisateur en fonction d'un taux de similarité.
Exemple 5 : La requête reformulée **Q_1'** pour la requête **Q_1** est (*Pesticide* **and** *Production).*

Définition8. (C*ube Pertinent CP*): un cube pertinent est le cube qui répond aux besoins de l'utilisateur. Un Cube Cj est pertinent si l'expression logique générée par Q, en remplaçant chaque terme *ti* par vrai s'il apparait comme instance pour au moins un composant structurel par faux sinon, est vraie.

Exemple 5: Le cube pertinent pour la requête reformulée **Q_1'** est le cube **C_2,** il contient les deux termes (*Pesticide et Production).*

5 Description de notre approche

Dans cette section nous présentons les étapes principales de notre approche (Figure 1) qui reçoit en entrée une requête de l'utilisateur et une collection de cubes de données et retourne en sortie un ensemble de cubes pertinents ordonnés. Elle est composée de trois grandes étapes : (1) Analyse de requête (i.e. définition des besoins décisionnels) via un ensemble de termes en langage naturel séparés par des AND et OR (Sec. 5.1), (2) recherche des termes dans les cubes via des index ad-hoc

(Sec. 5.2) et (3) attribution d'un ordre de pertinence pour les cubes trouvés à l'étape 2 (Sec. 5.3).

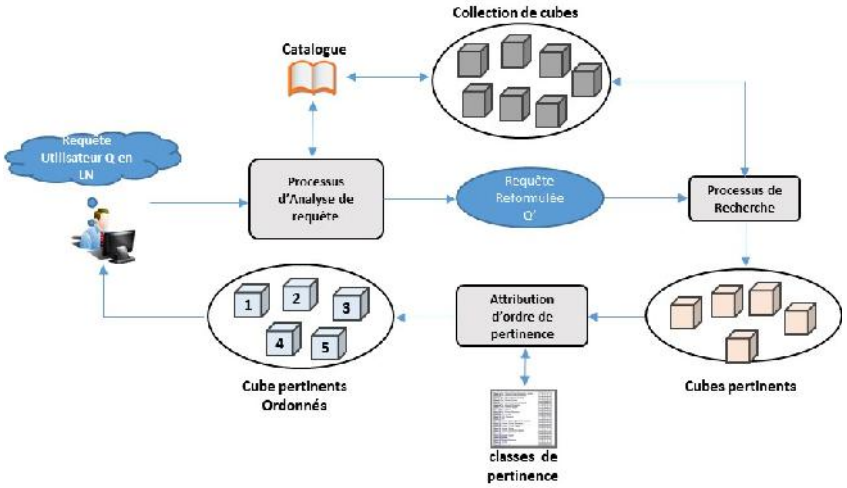


FIG. 1 – *Architecture générale de notre approche*

5.1 Processus d'analyse de Requêtes

Le décideur exprime généralement son besoin par une requête composée d'un ensemble de termes en langage naturel. Les termes de la requête sont séparés par des opérateurs logiques. Dans le cadre de ce travail, nous nous intéressons aux opérateurs AND et OR. Comme la collection de cubes peut être volumineuse et suggète à des mises à jour continuelles, l'utilisateur ne possède pas une vision claire et complète sur le contenu de la collection. Par conséquent, les termes utilisés dans sa requête peuvent ne pas être précis rendant le besoin exprimé ambigu et parfois obsolète. Afin de pallier ce problème, nous proposons d'effectuer une analyse de la requête qui consiste à la reformulation de la requête initiale. Ceci permet de faire un compromis entre le besoin de l'utilisateur exprimé en langage naturel et le contenu de la collection. Cela permet d'éviter les problèmes d'ambiguïté liés au langage naturel (Kuchmann-Beauger et Aufaure, 2011), ainsi que la difficulté de l'utilisation des langages informatiques complexes (comme SQL, MDX) par des décideurs non informaticien (Romero O., 2011). L'analyseur de requêtes que nous proposons, prend en entrée la requête initiale et retourne une autre requête reformulée. En effet, les termes saisis par l'utilisateur seront remplacés par des termes depuis la collection de cubes. Ces termes font référence aux instances des composants structurels (CS) des cubes décrits dans la Figure 2. En particulier, nous considérons qu'un cube est composé de plusieurs dimensions avec plusieurs niveaux. Un cube contient un fait qui est décrit par plusieurs mesures et chaque mesure est associée à plusieurs fonctions d'agrégation. Notons qu'une instance peut être simple ou

composé. Par exemple, *production* et *production laitière* sont deux instances différentes, la première simple et la deuxième est composée.

Par exemple pour le Cube *Production Laitière* (C_1), nous avons les instances suivantes, regroupées par composant structurel :

- *Dimensions* : opérateurs, produits, etc ;
- *Niveaux* : jour, mois, etc ;
- *Fait* : production laitière ;
- *Mesures* : quantité de lait produite, nombre animaux, etc ;
- *Agrégats* : total, moyenne, etc ;

Nous appelons « Catalogue » l'ensemble des instances (simples et composés) extraites à partir de

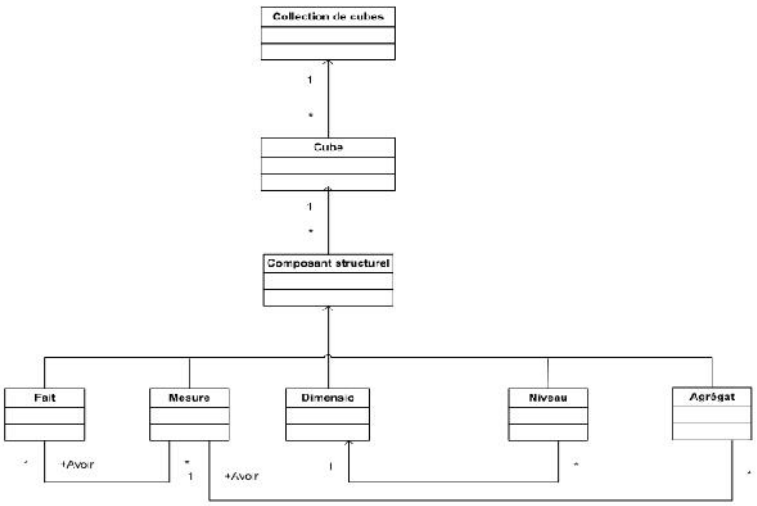


schéma OLAP de chaque cube de la collection.

FIG. 2 – *Un Méta modèle représentant les composants structurels (CS) de schéma OLAP*

4.2 Recherche des cubes pertinents

La recherche des cubes pertinents via les termes du Catalogue consiste à faire la recherche des termes de la requête reformulée dans les schémas OLAP de l'ensemble des cubes de la collection. A chaque requête de l'utilisateur, le système effectue un parcours dans tous les schémas OLAP des cubes de la collection. En général plusieurs utilisateurs peuvent effectuer plusieurs requêtes. D'autre part, dans une organisation on peut trouver une collection qui contient un nombre important de cubes. Cela posera un problème de temps à cause de parcours dans les schémas OLAP à chaque utilisateur et à chaque requête. Afin de palier à ce problème, nous proposons une approche de recherche basé sur les indexes.

Nous définissons deux types d'indexes, *indexes cubes* et *index collection*. La construction de l'index cube consiste à analyser pour chaque cube, ses composants structurels (CS). A partir des indexes cubes, un autre index, appelé *Index Collection*, est construit pour la collection de cubes. Nous présentons dans ce qui suit la structure et la construction de chaque index.

5.2.1 Index Cube

L'index cube est construit après une extraction des concepts depuis le schéma OLAP du cube. Il montre pour chaque terme trouvé, le composant structurel auquel (CS) il appartient : *fait, mesure, niveau, dimension* ou *agrégat*.

TAB. 3 – *Une partie représentative de l'index Cube Production Laitière*

	Fait	**Mesure**	**Dimension**	**Niveau**	**Agrégat**
Quantité du lait produite	0	1	0	0	0
Production laitière	1	0	0	0	0
Produits	0	0	1	0	0
Années	0	0	0	1	0
Production	0	0	1	0	0
Somme	0	0	0	0	1

L'index d'un cube C_m est représenté par une matrice IC_m contenant en ligne les termes ex- traits du cube, et en colonne les composants structurels. La matrice $IC_m(k_m X ||C S||)$, où k_m est le nombre de termes t_i trouvés dans le cube C_m et $CS = \{CS_F, CS_M, CS_D, CS_N, CS_A\}$ est l'ensemble des composants structurels. La matrice IC_m est remplie de la manière suivante (1):

$$IC_m [t_i,CS_j] = \begin{cases} 1 \text{ si le terme } t_i \text{ apparait comme valeur du composant structurel } CS_j \\ 0 \text{ sinon} \end{cases} \quad (1)$$

Par exemple, l'index cube C_2 «*production laitière* » de notre exemple de cas d'étude est représenté dans le tableau 3. Ce dernier montre, par exemple, la présence d'un fait *production laitière* dans ce cube.

5.2.2 Index Collection

La construction de l'index Collection *ICC* se fait à partir des index cubes, par l'union des termes de chaque cube dans la collection. Nous représentons l'index collection par une matrice *ICC* contenant en ligne les termes extraits, et en colonnes, les cubes de la collection (tableau 4).

La matrice *IC*(k_m X ||CS||) où *k* est le nombre de termes t_i extraits de la collection et ||C||, le nombre de cubes dans la collection *C*, est rempli de la manière suivante (2):

$$ICC[t_i,Cube_j] = \begin{cases} 1 \text{ si } \exists CS_k \text{ tel que } IC_j[t_i, CS_k]=1 \\ 0 \text{ sinon} \end{cases} \quad (2)$$

Lorsque l'utilisateur lance la recherche, une opération de mise en correspondance entre les termes sélectionnés par l'utilisateur à partir du catalogue et les termes contenus dans les index est effectuée. Le but étant de trouver les cubes pouvant constituer une réponse à la requête posée. La recherche s'effectue dans un premier temps sur l'index collection ensuite sur les indexes cubes.

TAB. 4 – *Une partie représentative de l'index Collection*

	Cube1	**Cube2**	**Cube3**
Quantité du lait produite	1	0	0
Nombre d'Animaux	1	0	0
Temps	1	1	1
Jour	1	1	1
Mois	1	1	1
Années	1	0	1
Production	1	1	1
Total	1	1	0
Moyenne	1	1	1
Concentration du Pesticide	0	1	0
Client	0	0	1
Pesticide	0	1	0

La recherche dans l'index collection renvoi les cubes correspondant aux termes recherchés (voir algorithme 1). L'algorithme identifie, pour chaque terme recherché, les cubes où il est référencé dans ICC. Il construit d'une manière incrémentale, l'ensemble des cubes où au moins un terme est référencé.

Algorithme 1 Recherche dans l'index collection

Input : Un besoin exprimé par un ensemble de termes $B\{t_1, \cdots, t_r\}$, une collection de n cubes $C_1, \cdots, C_n$, un index collection ICC.
Output : Un ensemble de cubes pertinents $CP = \{CP_1, CP_2, ..., CP_n\}$
Begin
CP :={} ;
  for each terme t_i dans B **do** Rechercher t_i dans ICC ;
    if (t_i existe) **then**
      for each cube C_k dans ICC **do**
        if $ICC[t_j, C_k] = 1$ **then**
          $CP = CP \cup \{C_k\}$
        end if
      end for
    end if
  end for
Return CP

End.

Pour illustrer cet algorithme, nous l'appliquons sur les requêtes *Q1* et *Q2* pour trouver les cubes pertinents. Pour *Q1*, l'algorithme retourne les cubes suivants :

- cube Production Laitière (C_1) avec le terme : production ;
- cube Production Agricole (C_2) avec les termes : production, pesticide ;
- cube Vente (C_3) avec le terme : production.

L'ensemble de cubes résultat est *CP*={C_1, C_2, C_3}. Pour *Q2*, l'algorithme retourne les cubes suivants :

- cube Production Laitière (C_1) avec les termes : quantité de lait produite, élevage, total ;
- cube Production Agricole (C_2) avec le terme : total ;
- cube Vente (C_3) avec le terme : total.

L'ensemble de cubes résultat est *CP*={C_1, C_2, C_3}. A partir de l'ensemble des cubes pertinents retournés par la recherche précédente, nous recherchons dans chaque cube le composant structurel où le terme t_i est référencé (voir algorithme 2). Le résultat de cette recherche retourne un ensemble de triplets T' <t_i,C_j ,CS_k >, où chaque triplet désigne le fait que le terme t_i apparait dans le cube C_j comme composant structurel CS_k .

Ces triplets sont regroupés par cubes et par composant structurel pour pouvoir calculer le nombre d'occurrences de chaque composant structurel dans chaque cube. Ainsi, après regroupement, nous associons à chaque cube C_j et chaque composant CS_k le nombre de fois où ils apparaissent. Un triplet résultat T a donc la forme suivante : <C_j , CS_k , $card_{jk}$ > qui est interprété par : le composant structurel CS_k est référencé $card_{jk}$ fois dans le cube C_j . La cardinalité calculée (le nombre d'occurrences) sera utilisée dans l'étape suivante pour classer les cubes pertinents ayant les mêmes composants structurels.

En appliquant cet algorithme aux requêtes *Q1*, *Q2*, nous obtenons le résultat suivant :

Pour *Q1* : cube Production Laitière (C_1) : (Dimension = *production*) ; cube Production Agricole (C_2) : (Dimension=*production*, Dimension=*pesticide*) ; cube Vente (C_3) : (Dimension=*production*). Les triplets résultats sont :

Algorithme 2 Attribution de l'ordre de pertinence

Input :un ensemble de cubes pertinents $CP = \{CP_1, CP_2, \cdots, CP_m\}$, un ensemble d'index cube
$IC_1, IC_2, \cdots, IC_n$., $B = \{t_1, \cdots, t_r\}$.
Output :un ensemble de triplets $T = \{< C_j, CS_k, Card_{jk} >\}$.
Begin
T'={ } // ensemble des triplets $\{< t_i, C_j, CS_k >\}$
for each Terme t_i dans B **do**
  for each Cube C_j dans CP_i **do**
    Rechercher les composants structurels CS_k où t_i est référencé dans IC_j
    for each CS_k trouvé **do**
      Générer les triplets $< t_i, C_j, CS_k > T' = T'U < t_i, C_j, CS_k >$
    end for
  end for
end for
// Calculer les cardinalités

```
for each cube C_j dans T' do
    Card_jk = 0;
    for each composant structurel CS_k dans T' do
        Card_jk = Card_jk + 1 ;// calculer le nombre d'occurrences de chaque CS_k
    end for
    Générer le triplet < C_j, CS_k, Card_jk > T = T ∪ < C_j, CS_k, Card_jk >
end for
 Return T
End
```

– T'={<production, C_1,dimension>, <production, C_2,dimension>, <pesticide, C_2,dimension>,
<production, C_3,dimension>} ;
– T={<C_1,dimension,1>, <C_2,dimension,2>, <C_3,dimension,1>}.

Pour *Q2* : cube Production Laitière (C_1) : (Mesure = *quantité de lait produite*, Dimension=*élevage*, Agrégat=*total*) ; cube Production Agricole (C_2) : (Agrégat=*total*) ; cube Vente (C_3) : (Agrégat=*total*). Les triplets en résultats sont :

– T'={< quantité de lait produite, C_1,mesure>, <élevage, C_1,dimension>, <total, C_1 ,agrégat>,
<total, C_2,agrégat>, <total, C_3,agrégat>} ;
– T={<C_1,mesure,1>, <C_1,dimension,1>, <C_1,agrégat,1>, <C_2,agrégat,1>, <C_3 ,agrégat,1>}.

5.3 Attribution d'ordre de pertinence

L'étape précédente identifie un ensemble de cubes correspondant aux termes définis par l'utilisateur. Ces cubes sont renvoyés à l'utilisateur sans aucun ordre préétabli. L'utilisateur doit rechercher le cube le plus pertinent par rapport à sa requête. Cette recherche devient de plus en plus complexe lorsque le nombre de cubes résultant de l'étape précédente est important. Pour aider l'utilisateur à choisir le cube qui répond le mieux à son besoin, il est nécessaire de définir des critères de pertinence permettant d'ordonner les cubes résultats. Notons que les termes définis par l'utilisateur peuvent apparaitre dans différents composants structurels d'un cube : *fait, mesure, dimension, etc*.

Pour pouvoir définir l'ordre des cubes, il faut définir le degré d'importance de chaque composant structurel dans la pertinence du cube. Il est évident que si les termes définis couvrent tous les composants structurels d'un cube, ce dernier est susceptible de mieux répondre à la recherche faite par l'utilisateur. Ce cube représente le sujet d'analyse recherchée, il contient la mesure (indicateur), les axes d'analyse (dimensions), l'information la plus fine pour les axes d'analyse (niveau), et la fonction de calcul (agrégat). En revanche, pour un cube conte- nant tous les composants structurels (*Fait*, *Mesure*, *Dimension*, *Niveau*) sauf l'agrégat, nous jugeons qu'il est moins pertinent que le précédent cube, où la fonction de calcule est couverte. Maintenant, si nous comparons entre *Mesure* et *Fait*, on peut dire que le sujet d'analyse (fait) est plus important que la mesure, parce qu'il est possible de trouver plusieurs cubes qui ont les mêmes mesures, mais avec des sujets d'analyse différents.

Entre *Dimension* et *Niveau*, l'information la plus fine est l'information la plus importante à l'utilisateur, car le niveau induit la dimension, mais le contraire n'est pas possible. Pour une dimension on peut trouver plusieurs niveaux, donc nous pouvons dire que le niveau est plus important que la dimension. Entre *Mesure* et *Dimension*, la mesure est plus importante que la dimension, du fait qu'une dimension (axe d'analyse) peut être retrouvée dans plusieurs cubes, et qu'elle sert à calculer plusieurs mesures dans plusieurs sujets d'analyses, même pour un seul cube. Nous considérons dans notre approche, qu'un cube contenant que des *agrégats* n'est pas considéré comme pertinent. Par exemple, l'agrégat *total* peut exister dans plusieurs cubes sans influencer sur la pertinence.

Compte-tenu de la discussion menée ci-dessus, nous pouvons proposer un ordre de pertinence des composants structurels. Cet ordre est défini, du plus pertinent vers le moins pertinent comme suit : *Fait*, *Mesure*, *Niveau*, *Dimension*, *Agrégat*. Nous voulons que notre approche ordonne les cubes d'une manière automatique, comme les moteurs de recherche classiques, selon l'ordre de pertinence par rapport au besoin de l'utilisateur.

Afin d'ordonner les cubes résultat C_1 , C_2 , … , C_k du plus pertinent au moins pertinent selon les composants CS_k couverts par les termes ti de l'ensemble B, nous proposons de définir l'ordre de pertinence sous forme de classes. Nous avons défini 22 classes de pertinence qui sont représentées dans la figure 3. Chaque classe est accompagnée par une codification utilisée par nos différents algorithmes.

Classe 1: Fait, Mesure [Agrégat], Niveau, Dimension	1 | 1 | 1 | 1 | 1
Classe 2: Fait, Mesure [Agrégat], Niveau	1 | 1 | 1 | 1 | 0
Classe 3: Fait, Mesure [Agrégat], Dimension	1 | 1 | 1 | 0 | 1
Classe 5: Fait, Mesure, Niveau, Dimension	1 | 1 | 0 | 1 | 1
Classe 6: Fait, Mesure, Niveau	1 | 1 | 1 | 1 | 0
Classe 7: Fait, Mesure, Dimension	1 | 1 | 0 | 1 | 0
Classe 4: Fait, Mesure [Agrégat]	1 | 1 | 1 | 0 | 0
Classe 8: Fait, Mesure	1 | 1 | 0 | 0 | 0
Classe 9: Fait, Niveau, Dimension	1 | 0 | 0 | 1 | 1
Classe 10: Fait, Dimension	1 | 0 | 0 | 0 | 1
Classe 11: Fait	1 | 0 | 0 | 0 | 0
Classe 12: Mesure [Agrégat], Niveau, Dimension	0 | 1 | 1 | 1 | 1
Classe 13: Mesure [Agrégat], Niveau	0 | 1 | 1 | 1 | 0
Classe 14: Mesure [Agrégat], Dimension	0 | 1 | 1 | 0 | 1
Classe 16: Mesure, Niveau, Dimension	0 | 1 | 0 | 1 | 1
Classe 17: Mesure, Niveau	0 | 1 | 0 | 1 | 0
Classe 18: Mesure, Dimension	0 | 1 | 0 | 0 | 1
Classe 15: Mesure [Agrégat]	0 | 1 | 1 | 0 | 0
Classe 19: Mesure	0 | 1 | 0 | 0 | 0
Classe 20: Niveau, Dimension	0 | 0 | 0 | 1 | 1
Classe 21: Niveau	0 | 0 | 0 | 1 | 0
Classe 22: Dimension	0 | 0 | 0 | 0 | 1

FIG. 3 – *Les classes de Pertinences*

Nous proposons trois étapes pour ordonner les cubes pertinents : (1) trouver la classe d'appartenance de chaque cube C_j dans *CP*, (2) ordonner les classes trouvées selon l'ordre de pertinence que nous venons de proposer et (3) ordonner les cubes de la même classe selon la cardinalité du composant structurel. Algorithme 3, décrit notre approche d'attribution de l'ordre de pertinence. Pour trouver la classe de pertinence de chaque cube, on identifie dans *T*, l'ensemble des composants structurels lui correspondant. Cet ensemble sera codé de la même manière que celle utilisée dans

figure 2. Par exemple, à partir de l'ensemble *T* de la requête *Q2* (<C_1,mesure,1>, <C_1 ,dimension,1>, <C_1 ,agrégat,1>, <C_2 ,agrégat,1>, <C_3 ,agrégat,1>), nous identifions pour le cube C_1, l'ensemble des composants suivants : *mesure*, *dimension* et *agrégat*. Cet ensemble, sera codé comme suit : 0 | 1 | 0 | 1 | 1 |. Pour les besoins de nos algorithmes, nous définissons la fonction *GetCode* (C_i) qui génère ce code pour le cube C_i.

En attribuant l'ordre de pertinence sur le résultat de recherche des requêtes *Q1* et *Q2*, nous obtenons le résultat suivant : Pour *Q1* (pesticides par production), à partir des cubes pertinents de l'étape précédente *CP*={C_1, C_2, C_3} où C_1, C_2 et C_3 sont respectivement les cubes *Production Laitière*, *Production Agricole* et *Vente*, le résultat final est *CF*={C_1, C_2, C_3}.

Algorithme 3 Attribution de l'ordre de pertinence

Input : Ensemble de cubes pertinents $CP = \{CP_1, CP_2, ..., CP_m\}$, un ensemble de triplets retournés par l'étape de recherche $T < C_j, CS_k, Card_{jk} >$, l'ensemble des classes de pertinences Cls,
Output : Les cubes pertinents dans l'ordre décroissant $CF = \{CF_1, CF_2, ..., CF_n\}$;
Begin
CF :=CP ;
for each cube Cj de CF **do**
for each classe Cl in Cls **do**
if GetCode (Cj) = Code(Cl) **then**
Classe (Cj)= Cls ;
end if
end for
end for
Ordonner CF selon l'ordre des classes obtenues ;
//Ordonner les cubes appartenant à la même classe
for each cube Ci dans CF **do**
for each cube Cj dans CF **do**
if Classe(Ci) = Classe (Cj) **then**
Mettre à jour l'ordre selon la cardinalité dans l'ensemble T ;
end if
end for
end for
Return CF
End.

Pour cette requête, l'utilisateur veut analyser *les pesticides par production*. Sémantiquement, le cube qui répond à ce besoin serait le cube *production agricole*. Notre approche retourne ce même cube comme le cube le plus pertinent. Il contient deux dimensions, tandis que les cubes production laitière et Ventes comportent chacun une dimension.

Pour Q2 (total de la quantité de lait produite par élevage), *CP*={C_1, C_2, C_3}, le résultat final est *CF*={C_1}. Les cubes C_2 et C_3 ont été éliminés puisqu'ils ne contiennent que des agrégats. Si nous analysons sémantiquement le besoin de l'utilisateur dans *Q2*, nous constatons que le cube production laitière répond à ce besoin, car il permet d'analyser la quantité produite du lait. Notre approche retourne ce même cube pour *Q2*. Ce cube contient une mesure, une dimension et un agrégat. Les deux autres cubes (production agricole et vente) ne contiennent que des agrégats donc aucune donnée permettant de répondre au besoin exprimé dans *Q2*.

6 Implémentation et Evaluation

6.1 Implémentation

Afin de valider notre approche de recherche de cubes de données, nous avons réalisé un outil appelé ***Cube-Finder***. Cet outil permet de trouver parmi plusieurs cubes de données le/les cube(s) pertinent(s) pour répondre à un besoin utilisateur. Le résultat est ensuite renvoyé trié selon un ordre de pertinence.
Avant de présenter notre outil, nous présentons d'abord son architecture en quatre tiers (voir Figure 4). Elle est composée de quatre niveaux : niveau serveur OLAP *Mondrian*, niveau serveur de données *PostgresSql*, niveau de visualisation (notre outil *Cube-Finder*) et niveau manipulation *Juribik*. Pour la réalisation de notre outil *Cube-Finder*, nous avons utilisé une solution complètement opensource basée sur *PostgresSql* comme SGBD, *Mondrian* comme serveur OLAP et *Juribik* comme interface de manipulation des cubes.

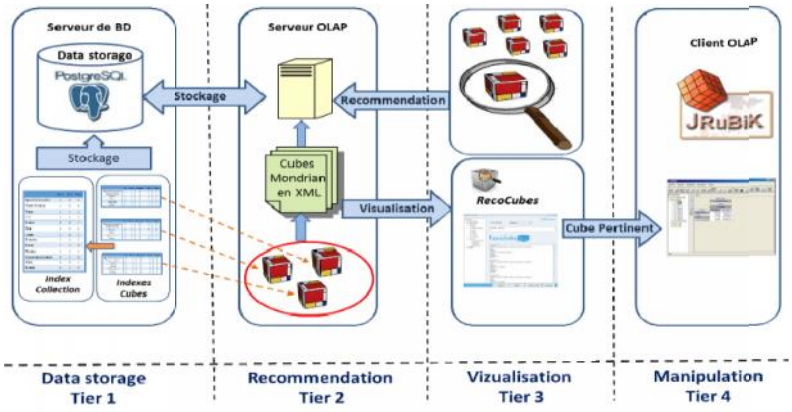
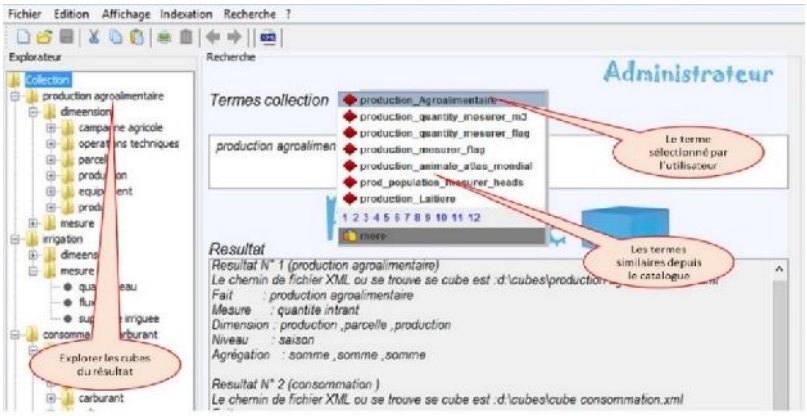


FIG. 4 – *Architecture de notre environnement*

L'outil développé *Cube-Finder*, comprend deux interfaces, une pour l'administrateur et l'autre pour les utilisateurs. La première permet à l'administrateur de charger les schémas OLAP des cubes (en *XML*), d'indexer les cubes, de les explorer et de faire la recherche. La deuxième interface offre à l'utilisateur un espace très convivial pour rechercher les cubes de données. L'utilisateur sélectionne les termes depuis le catalogue, l'outil affiche le résultat de la recherche triée selon l'ordre de pertinence. L'utilisateur peut explorer ou naviguer dans les cubes résultats pour affiner sa recherche. Dans cette interface (Figure 5), nous présentons le résultat de la requête



Q2 de notre cas d'étude.

FIG. 5 – *L'interface réservée à l'utilisateur dans notre outil Cube-Finder*

6.2 Evaluation

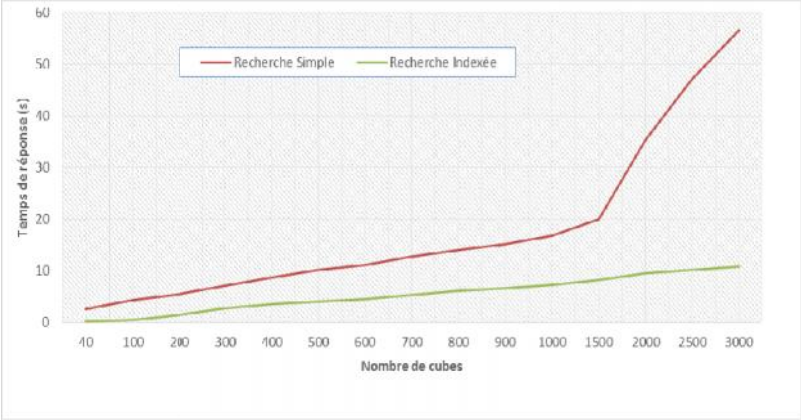
Après avoir réalisé notre outil de recherche des cubes pertinents (*Cube-Finder*), nous avons effectué des tests pour évaluer notre approche.
Nous présentons dans cette section, une série d'expériences que nous avons effectuées pour valider notre approche. Les tests effectués concernent le temps de réponse (performance) et la pertinence des résultats.
Pour les besoins de nos expériences, nous considérons un dataset simulé composé de 3000 cubes obtenus en dupliquant 50 cubes dans le domaine agricole (d'une ferme pilote) et de site DataFao[4]. Nous avons effectué les tests sur une machine dell Inspiron avec 4Go de ram et un Processeur i5.

6.2.1 Evaluation de performance

Afin de d'évaluer les performances de notre approche, nous avons effectué deux types de recherches : avec indexes et sans indexes (effectuer la recherche en parcourant tous les schémas OLAP). Nous avons réalisé trois types de tests, selon le nombre de cubes de notre collection et selon le nombre de termes de la requête.

6.2.1.1 Evaluation du temps de réponse en fonction du nombre de cubes

Afin d'évaluer le temps de réponse en fonction du nombre de cubes, nous avons calculé le temps de réponse d'une recherche simple et d'une recherche avec indexes pour une collection de 3000 cubes et une charge de 8 requêtes composée chacune entre 2 et 5 termes. Nous avons calculé la moyenne du temps de réponse des 8 requêtes. Nous considérons qu'un utilisateur peut interroger la collection de cubes plusieurs fois par jours (une fois par heure dans nos tests).



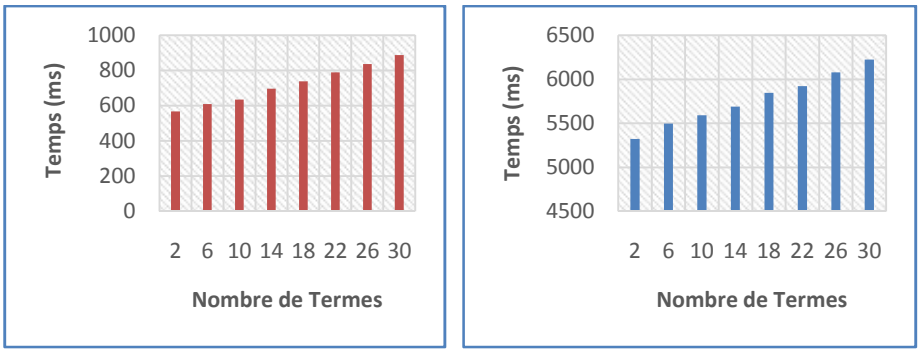
[4] data.fao.org

FIG. 6 – *Evaluation du temps de réponse selon le nombre de cubes*

La Figure 6 montre l'évaluation du temps de réponse en fonction du nombre de cubes. Les résultats obtenus montrent que lorsque le nombre de cubes est réduit le gain de performance de la recherche avec indexes par rapport à la recherche simple n'est pas assez important. Lorsque le nombre de cubes augmente le gain de performance devient de plus en plus important. Cela est dû aux temps consommé pour la récupération des composants structurels des cubes à partir de la collection (parcours des schémas OLAP des cubes pour chaque requête utilisateur). Dans la recherche avec index, nous récupérons le contenu de l'index une seule fois et la recherche sera effectuée dans l'index. Nous pouvons conclure que l'utilisation des index est très utile dans le cas d'une interrogation massive de la collection des cubes par un ou plusieurs utilisateurs.

6.2.1.2 Evaluation du temps de réponse en fonction du nombre de termes

La requête utilisateur comporte un ou plusieurs termes. Afin de voir l'influence du nombre de termes de la requête sur le temps de réponse, nous avons effectué des tests sur une collection de 3000 cubes et une charge de 29 requêtes ayant de 2 jusqu'à 30 termes. Nous obtenons les résultats suivants :



(a) – Recherche avec Indexes (b) – Recherche Simple

FIG. 7 – *Temps de Réponse vs. Nombre de termes*

La Figure (7-a) et (7-b) montre l'évaluation de la performance en fonction de nombre de termes dans la requête utilisateur. Les résultats montrent que lorsque le nombre de termes augmente le temps de réponse augmente quel que soit le type de recherche utilisé. Cela est dû à l'augmentation du nombre de vérification à faire pour chaque requête.

6.2.2 Evaluation de pertinence

La qualité d'un système doit être mesurée en comparant les réponses du système avec les réponses idéales que l'utilisateur espère recevoir. Plus les réponses du système correspondent à celles que l'utilisateur espère, mieux est le système.
Dans cette section, nous évaluons la qualité des cubes retournés par notre approche. Pour cela, nous considérons deux paramètres, le rappel (3) et la précision (4).

$$Rappel = \frac{\text{Nombre de cubes pertinents retournés par le système}}{\text{Nombre de cubes pertinents donnés par l'expert}} \quad (3)$$

$$Précision = \frac{\text{Nombre de cubes pertinents retournés par le système}}{\text{Nombre de cubes retournés par le système}} \quad (4)$$

Nous considérons un ensemble de 50 cubes et trois catégories de Requêtes : (1) Requêtes Ciblée (*Requête_AND)* : requête composée de plusieurs termes séparés par l'opérateur ***AND*** ; (2) Requête Vague (*Requête_OR)*: requête composée de plusieurs termes séparés par l'opérateur ***OR***; (3) *Requête_AND_OR*: Requête composée par plusieurs termes séparés par des opérateurs ***AND*** et ***OR***. Pour chaque catégorie, nous avons défini 36 requêtes, 4 requêtes par nombre de termes utilisés : de 2 à 10 termes.

Après l'affichage des cubes pertinents par l'outil, le décideur s'intéresse généralement aux meilleurs premiers cubes du résultat. Pour évaluer la qualité des cubes retournés nous avons considéré les 10 premiers cubes du résultat. Nous calculons pour chaque catégorie de requêtes la précision et le rappel.

a) *Précision* (b) *Rappel*

FIG. 8 – L'*Evaluation de pertinence*

Les résultats de la Figure 8 montrent que notre approche est intéressante en termes de précision concernant les premiers cubes. A partir du 3ème cube la précision est élevée pour les requêtes ciblées (*Requête_OR_AND* et *Requête_AND)* par rapport aux requêtes vagues (*Requête_OR)*. Notre système renvoie entretemps des cubes non pertinents (bruit) pour les requêtes vagues plus que les requêtes précises. En termes de rappel, le système donne un taux plus élevé pour les requêtes vagues par rapport aux requêtes précises pour les premiers cubes. Autrement dit, notre approche donne un

bon rappel à partir du 5èmes cube où les cubes retournés coïncident avec tous les cubes jugés pertinents par l'expert.

6 Conclusion et travaux futurs

Dans ce travail, nous avons présenté une approche de recherche de cubes de données. Cette approche permet de trouver parmi plusieurs cubes de données le ou les cube(s) qui répondent le mieux à un besoin utilisateur. Nous avons réalisé un outil de recherche de cubes de données pertinents. Cet outil offre une interface conviviale pour l'utilisateur et l'administrateur. Nous avons proposé un ordre de pertinence pour classer les cubes de données résultats. Cet ordre est défini par rapport à l'existence des termes sélectionnés par l'utilisateur dans les différents composants structurels des cubes (*fait*, *mesure*, *dimension*, *niveau*, *agrégat*). Actuellement, nous améliorons notre outil de recherche de cubes en y intégrant une deuxième recherche au niveau SGBD afin d'avoir l'information la plus fine par l'ajout d'un composant structurel membre. Nous travaillons sur une approche de construction de cubes qui permet de construire des cubes avec des besoins dispersés sur plusieurs cubes en réponse à une requête de l'utilisateur.

Références

Berro, A., I. Megdiche, et T. O (2013). Vers l'intégration multidimensionnelle d'open data dans les entrepôts de données. In *EDA 2013*.

Bimonte, S., K. Boulil, J.-P. Chanet, et M. Pradel (2012). Definition and analysis of new agricultural farm energetic indicators using spatial olap. In *ICCSA (2)*, pp. 373–385.

Cabibbo, L. et R. Torlone (2004). On the integration of autonomous data marts. In *16th International Conference on Scientific and Statistical Database Management ACM Press*.

Feki, J., H. Ben-Abdallah, et M. Ben Abdallah (2006). Réutilisation des patrons en étoile. In *INFORSID 2006*. Giacometti, A., P. Marcel, et E. Negre (2008). A framework for recommending olap queries. In *Proceedings of the ACM 11th international workshop on Data warehousing and OLAP*.

Golfarelli, M., Rizzi, S. and Biondi, P.: myOLAP: An Approach to Express and Evaluate OLAP Preferences. IEEE Trans. Knowl. Data Eng. 23(7): 1050-1064 (2011)

Jerbi, H., G. Pujolle, F. Ravat, et O. Teste (2011). Recommandation de requêtes dans les bases de données multidimensionnelles annotées. In *Ingénierie des systèmes d'Information*.

Khemiri, R. and Bentayeb , F.: FIMIOQR: Frequent Itemsets Mining for Interactive OLAP Query Recommendation. In DBKDA 2013, The Fifth International Conference on Advances in Databases, Knowledge, and Data Applications.

Koutrika, G., B. Bercovitz, et H. Garcia-Molina (2009). Flexrecs : Expressing and combining flexible recommendations. In Proceedings of the 2009 ACM SIGMOD International Conference on Management of data (SIGMOD).

Kozmina, N.: Adding Recommendations to OLAP Reporting Tool. ICEIS (1) 2013: 169176

Kuchmann-Beauger, N. et M. Aufaure (2011). A natural language interface for data warehouse question answering. In 16th International Conference on Applications of Natural Language to Information Systems.

Negre, E. (2009). Recommandations personnalisées de requêtes mdx. In 5èmes journées francophones sur les Entrepôts de Données et l'Analyse en ligne, EDA 2009, pp. 23–36.

Romero O., A. A. (2011). Multidimensional design methods for data warehousing integrations. In Data Mining and Database Technologies.

Umagandhi, R., Senthil Kumar A. V.: Time heuristics ranking approach for recommended queries using search engine query logs. In: Kuwait Journal of Science and Engineering. vol. 41, pp. 127-149, 2014.

Wael H. Gomaa, Aly A. Fahmy: A Survey of Text Similarity Approaches. IN International Journal of Computer Applications (0975 – 8887). Volume 68– No.13, April 2013.