Réduction de la quantité de données à visualiser dans l'OLAP multifonctions

Ali HASSAN[1], Patrice DARMON[1]

1. équipe R&D, Umanis
7-9 rue Paul Vaillant Couturier, 92300, Levallois-Perret, France
ahassan@umanis.com, pdarmon@umanis.com

Résumé. Dans le contexte d'OLAP multifonctions, le modèle conceptuel de l'entrepôt de données permet d'associer à une même mesure, une fonction d'agrégation différente pour chaque axe d'analyse, chaque hiérarchie et chaque niveau de granularité. Un système OLAP permet d'analyser les données multidimensionnelles interactivement à l'aide de tableaux croisés dynamiques et diagrammes. Malheureusement, aucun travail n'étudie la question de la lisibilité des diagrammes dans le contexte d'OLAP multifonctions. Donc, dans cet article, nous proposons une méthode de post-traitement des résultats des requêtes OLAP multifonctions afin d'améliorer la lisibilité en réduisant la quantité de données à visualiser. Cette méthode consiste à agréger les données vers des granularités moins détaillées. Autrement dit, faire un forage vers le haut (Rollup). La méthode proposée exploite d'abord le modèle conceptuel de l'entrepôt de données afin de trouver les agrégations qui ont été déjà réalisées dans la requête OLAP. Ensuite, à partir du résultat de la requête OLAP, en contrôlant la validité des fonctions d'agrégation, cette méthode trouve les opérations Rollup possibles et élimine celles qui sont interdites. Finalement, la méthode analyse les opérations Rollup possibles pour en choisir une qui donne à la fois des diagrammes lisibles et garde le maximum de détails possibles.

ABSTRACT. In the context of multifunction OLAP, the data warehouse conceptual model allows to associate to the same measure, a different aggregation function for each analysis axis, each hierarchy and each granularity level. An OLAP system allows to analyze interactively multidimensional data using pivot tables and diagrams. Unfortunately, no works investigate readability issues of diagrams in the context of multifunction OLAP. Therefore, in this article, we propose a method of post-processing of multifunction OLAP queries results to improve readability by reducing the amount of data to be visualized. This method aggregates the data to less detailed granularities. In other words, doing a Rollup. The proposed method first exploits the conceptual data warehouse model to find aggregations that have already been done in the OLAP query. Then, from the OLAP query result, by checking the aggregation functions validity, this method finds possible Rollup operations and eliminates those that are forbidden. Finally, the method analyzes possible Rollup operations to select the one that both gives readable diagrams and keeps as many details as possible.

Mots-clés : visualisation de données, OLAP, agrégation multifonctions, réduction de données.

KEYWORDS: data visualization, OLAP, multifunction aggregation, data reduction.

1. Introduction

L'analyse OLAP consiste à observer des indicateurs (mesures) selon plusieurs axes d'analyse (dimensions). Les décideurs utilisent des opérateurs OLAP (*Rollup* et *Drill-down*) afin d'analyser les mesures selon plusieurs niveaux de détails. Les données sont donc regroupées selon les niveaux sélectionnés et agrégés en utilisant des fonctions d'agrégation. Classiquement, l'OLAP offre la possibilité d'utiliser une seule fonction pour agréger une mesure dans tout l'espace multidimensionnel. Afin de surmonter cette limitation, *Hassan et al.* (2014) ont proposé un modèle assez expressif pour associer plusieurs fonctions d'agrégation à la même mesure. Par exemple, un tel modèle permet d'analyser la précipitation annuelle qui est calculée par la **somme** des précipitations quotidiennes. Cependant, la précipitation moyenne d'un département est calculée par la **moyenne** des précipitations de toutes les villes. Comme les fonctions généralement ne sont pas commutatives, le modèle proposé contrôle la validité du calcul en planifiant un ordre d'exécution entre les fonctions d'agrégation. Par exemple, il faut agréger les précipitations par la somme (avec un ordre d'exécution 1) sur la dimension 'Date' avant de les agréger par la moyenne (avec un ordre d'exécution 2) sur la dimension 'Géographie'.

Le résultat d'une requête OLAP est visualisé classiquement en utilisant un tableau croisé dynamique et des diagrammes (en barres, en secteurs, etc.). Par exemple, le tableau croisé dynamique en haut de la figure 1 montre la précipitation moyenne quotidienne (les deux premiers jours de janvier et février) par département. On peut apercevoir qu'une telle table a une lisibilité acceptable bien qu'elle affiche 120 données. L'affichage de même nombre de données en diagramme en barres peut réduire considérablement la lisibilité (figure 1 bas à gauche) ou voir devenir complètement illisible en utilisant un diagramme en secteurs (figure 1 bas à droite). Notre objectif est de proposer un système qui permet d'adapter automatiquement la quantité de données à afficher afin de garder la lisibilité des diagrammes utilisés.

Précipitation moyenne		Année	2010				2011				2012				2013				2014			
		Mois	janv		févr		janv		févr		janv		févr		janv		févr		janv		févr	
		jour	1	2	1	2	1	2	1	2	1	2	1	2	1	2	1	2	1	2	1	2
Pays	Région	Département																				
France	Midi-Pyrénées	Haute-Garonne	12,38	13,75	13,75	12,88	11,63	11,38	11,63	0	14,38	13,5	12,38	11,38	11,88	13,38	11,75	11,38	13,63	14,13	11,63	11,25
		Hautes-Pyrénées	11,88	10,63	10,88	12,38	14,38	12,25	15,38	12	8,88	10	13,25	9	12,13	10,5	10,75	16,38	11,75	12,13	11,63	14
	île-de-france	Paris	9,5	13	13,13	13,25	12,38	8,75	14,25	10,38	13,13	11,5	13,5	11,63	10,25	11,75	10,63	12,88	14,25	12,25	13,75	13,75
		Seine-et-Marne	12,75	11,25	12	11,38	13,75	11,88	13,5	13,5	11,5	14,38	10,75	12	12,38	12	13,5	13,88	15,13	13,63	13,25	12,88
	Rhône-Alpes	Rhône	11,63	11,38	12,5	11,88	11,25	12,75	16,13	14	11,38	11,63	11,63	13,38	10,75	12,38	11,63	13,25	11,38	13,13	11,25	12,5
		Isère	12,88	10,75	9,63	10,13	13,13	12,88	8,5	9,63	11,75	13,13	13,13	12,25	12	11,38	10,63	12,63	12,13	9,88	14,5	10,13

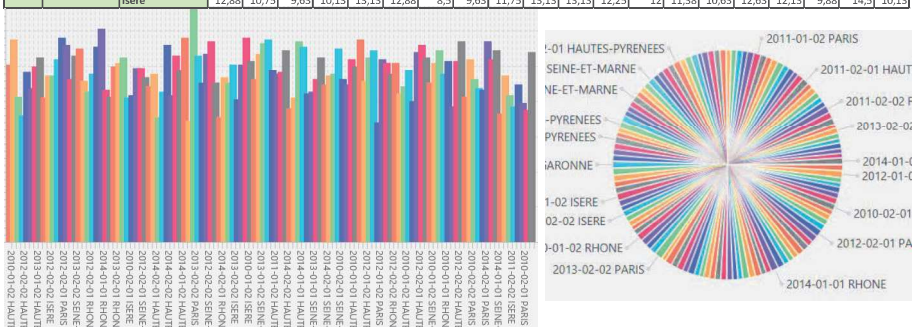

Figure 1. Visualisation des précipitations moyennes par jour et département

2. État de l'art

La plupart des travaux existant concernant la visualisation de grandes quantités de données (Wang *et al.*, 2017 ; Pahins *et al.*, 2017 ; Miranda *et al.*, 2018) ont pour objectif d'améliorer la performance du système en pré-agrégeant les données et en stockant les résultats en mémoire. La lisibilité des données visualisées a été traitée dans certains travaux. Les travaux : (Silva, *et al.*, 2012) pour les données spatiales, (Jugel *et al.*, 2014) pour les données temporelles et (Golfarelli *et al.*, 2014) pour la lisibilité du tableau croisé dynamique, proposent de réduire la quantité de données visualisées en agrégeant et regroupant (clustering) les données. Malheureusement, aucun de ces travaux n'étudie comment réduire la quantité de données dans le contexte d'OLAP multifonctions. Donc, dans cet article, nous proposons une méthode pour améliorer la lisibilité des résultats des requêtes OLAP multifonctions.

3. Méthode de réduction des données

Notre proposition est basée sur une méthode de post-traitement des résultats des requêtes OLAP multifonctions. Cette méthode réduit la quantité de données à visualiser en fonction du diagramme utilisé. Par exemple, les diagrammes en barres peuvent visualiser normalement plus de données que les diagrammes en secteurs. Ce qui veut dire que les diagrammes en secteurs nécessitent d'agréger (réduire) les données plus que les diagrammes en barres.

Cette méthode consiste à agréger les données automatiquement vers des granularités moins détaillées. Autrement dit, faire un forage vers le haut (Rollup). Cette méthode comprend plusieurs étapes :

1- Étudier la requête OLAP courante : notre méthode exploite le modèle de l'entrepôt de données afin de trouver l'ordre d'exécution maximal des fonctions d'agrégation qui ont déjà été réalisées au cours de la requête. La requête se représente par la mesure et les niveaux de granularité de chaque dimension sélectionnée pour observer la mesure. Par exemple, la requête de la figure 1 peut se représenter par la mesure 'précipitation moyenne' et les niveaux ['Jour', 'Département'].

La méthode cherche les fonctions d'agrégation utilisées pour agréger la mesure à chaque niveau de granularité de la requête courante. Une fois que ces fonctions sont trouvées, la méthode trouve l'ordre d'exécution maximal. Dans notre exemple de la figure 1, le niveau 'Jour' est le niveau le plus détaillé sur la dimension 'Date', donc il n'y a pas d'agrégation sur cette dimension. Sur la dimension 'Géographie', la précipitation est agrégée au niveau 'Département' à partir de 'Ville' par la moyenne qui a un ordre d'exécution 2 qui est donc l'ordre d'exécution maximal déjà réalisé.

2- Trouver les opérations Rollup possibles : afin de réduire la quantité de données à visualiser, nous proposons de réaliser un Rollup à partir du résultat de la requête sans interroger la base de données. En contrôlant la validité de calcul des fonctions d'agrégation, notre méthode trouve les opérations Rollup possibles et élimine celles qui sont interdites. Nous pouvons déterminer les opérations Rollup possibles par le produit cartésien de listes de niveaux de granularité possibles sur

chaque dimension. Un niveau de granularité est considéré comme "possible" si la condition d'ordre d'exécution est satisfaite. Selon cette condition, la fonction d'agrégation qui agrège la mesure au niveau concerné devrait avoir un ordre d'exécution égal ou supérieur à l'ordre d'exécution maximal déjà réalisé. Ainsi, notre méthode examine tous les niveaux supérieurs aux niveaux actuels de la requête.

Concernant la figure 1, l'ordre d'exécution maximal déjà réalisé est 2. Donc, les niveaux ('Région' et 'Pays') sur la dimension 'Géographie' sont possibles parce que la mesure est agrégée à ces niveaux par la moyenne qui a un ordre d'exécution 2 contrairement aux niveaux ('Mois' et 'Année') sur la dimension 'Date' qui sont interdits parce que la mesure est agrégée à ces niveaux par la somme qui a un ordre d'exécution 1. On ajoute les niveaux actuels de la requête ('Département' et 'Jour') aux listes de niveaux possibles correspondant avant de faire le produit cartésien. Ainsi, les listes de niveaux possibles sur les dimensions 'Date' et 'Géographie' sont respectivement ['Jour'] et ['Département', 'Région', 'Pays']. On enlève l'analyse actuelle ['Jour', 'Département'] du résultat du produit cartésien de ces listes. Donc, les opérations Rollup possibles trouvées sont ['Jour', 'Région'] et ['Jour', 'Pays'].

3- Calculer la taille des données pour toutes les opérations Rollup possibles : pour calculer la taille des données pour une opération Rollup, il suffit de multiplier le nombre de membres de ses niveaux de granularité. Nous pouvons trouver le nombre de membres de niveaux de granularité en analysant les en-têtes (en ligne et en colonne) du tableau croisé dynamique du résultat de la requête de base. Par exemple, en regardant la figure 1, nous constatons qu'il y a 20 jours (20 membres), 3 régions et 1 pays. Ainsi, la taille des données de deux opérations Rollup possibles résultant de l'étape précédente est calculée comme suit :

– La taille des données de ['Jour', 'Région'] = 20 × 3 = 60

– La taille des données de ['Jour', 'Pays'] = 20 × 1 = 20

4- Choisir une opération Rollup : notre méthode réduit la taille des données selon le type de diagramme utilisé (par exemple, diagrammes en barres ou en secteurs). Elle est contrôlée par des paramètres exprimant la taille maximale autorisée pour chaque type de diagramme. Comme la question de la lisibilité des données est relative (c'est-à-dire que ce qui est lisible pour une personne ne l'est pas forcément pour tout le monde), nous supposons que ces limites sont définies par l'utilisateur lui-même. Notre méthode choisit automatiquement une opération Rollup pour chaque type de diagramme qui à la fois donne un diagramme lisible (respecte la taille maximale autorisée) et garde le maximum de détails possible.

Par exemple, un utilisateur pourrait déterminer la taille maximale des diagrammes en barres et en secteurs par 70 et 25 respectivement. En d'autres termes, un diagramme en barres peut avoir au maximum 70 barres et un diagramme à secteurs peut avoir jusqu'à 25 secteurs. Dans ce cas, notre méthode sélectionne l'agrégation des données au niveau ['Jour', 'Région'] pour le diagramme en barres et au niveau ['Jour', 'Pays'] pour le diagramme en secteurs.

5- Réaliser l'opération Rollup choisie : une fois que les agrégations (les opérations Rollup) sont sélectionnées, elles doivent être exécutées. Ainsi, la méthode

devrait trouver les fonctions d'agrégation qui agrègent la mesure sur toutes les dimensions entre les niveaux actuels et les niveaux de l'opération Rollup choisie. Ensuite, les fonctions trouvées doivent être exécutées en fonction de leur ordre d'exécution. Si la fonction d'agrégation est algébrique, des valeurs intermédiaires doivent être stockées. Par exemple, pour éviter le calcul de la moyenne des moyennes, la fonction algébrique AVG nécessite de stocker les valeurs intermédiaires SUM et COUNT. Finalement, les diagrammes visualisent les données agrégées (cf. figure 2).

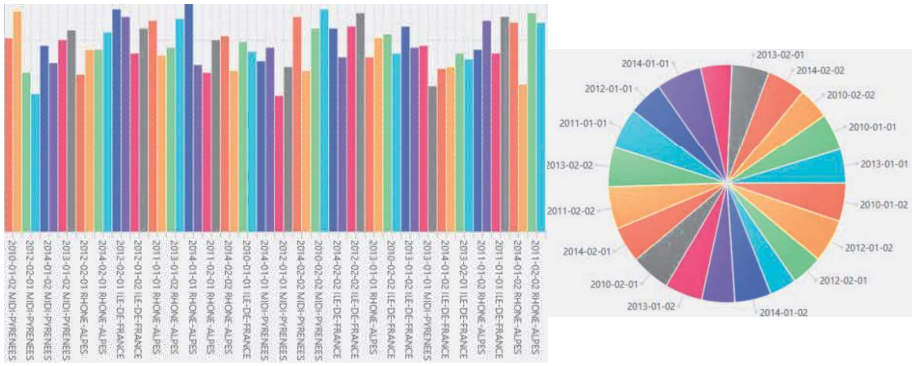

Figure 2. Visualisation des précipitations moyennes améliorée

4. Implémentation

Dans cette section, nous démontrons la faisabilité de notre proposition. Notre méthode est implémentée comme une extension du prototype 'OLAP-Multi-Functions' (*Hassan et al.,* 2015). Ce prototype a une architecture à deux niveaux (Figure 3). Le premier est les *interfaces*. Il est développé en Java. Il permet de créer et visualiser les schémas multidimensionnels multifonctions et d'interroger l'entrepôt de données. Le deuxième est le *stockage,* où un SGBDR (Oracle), est utilisé pour stocker le méta-schéma et l'entrepôt de données. Un générateur de requêtes SQL est développé en tant que procédures stockées. Il traduit, en prenant en compte le contexte multifonctions, les interactions des utilisateurs (requêtes OLAP) en requêtes SQL.

Notre méthode est développée en Java. Elle prend en entrée deux fichiers ('Données.JSON' et 'Mondrian.xml'). Le premier est le résultat de la requête OLAP en format JSON. Le deuxième est une extension du schéma Mondrian où nous décrivons l'association de plusieurs fonctions d'agrégation à la même mesure.

Les cinq étapes de la méthode sont exécutées séquentiellement (l'une après l'autre). Si plusieurs types de visualisation sont utilisés (par exemple, en barres, en secteurs), une exécution multiple des étapes 4 (choisir une opération Rollup) et 5 (réaliser l'opération Rollup choisie) est nécessaire. Ainsi, une exécution parallèle (un thread par type de visualisation) est utilisée. Les résultats de la réduction de données sont stockés par type de visualisation (par exemple, les fichiers 'Barre.JSON' et 'Secteur.JSON') avant d'être envoyés à la visualisation correspondante.

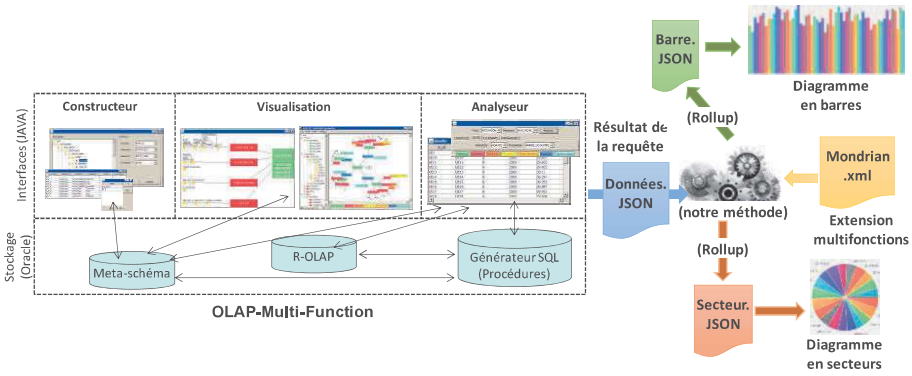

Figure 3. Implémentation de la méthode (Prototype)

5. Conclusion

Nous avons présenté une méthode de post-traitement pour améliorer la lisibilité des requêtes OLAP multifonctions. Elle permet la réduction automatique de la quantité de données selon le diagramme utilisé. Elle prend en compte la validité de calcul afin d'éviter les opérations d'agrégation de données (Rollup) interdites.

Bibliographie

Golfarelli M., Graziani S., Rizzi S. (2014). Shrink: An OLAP operation for balancing precision and size of pivot tables. *Data and Knowledge Engineering*, vol. 93, p. 19-41.

Hassan A., Ravat F., Teste O., Tournier R., Zurfluh G. (2014). Multidimensional database modelling with differentiated multiple aggregations. *JDS*, vol. 23, n° 4, p. 437-459.

Hassan A., Ravat F., Teste O., Tournier R., Zurfluh G. (2015). Differentiated Multiple Aggregations in Multidimensional Databases. *TLDKS*, vol. 21, p. 20-47.

Jugel U., Jerzak Z., Hackenbroich G. (2014). M4 : A visualization-oriented time series data aggregation. *PVLDB*, vol. 7, n° 10, p. 797–808.

Miranda F., Lins L., Klosowski J. T., Silva C. T. (2018). Topkube: A rank-Aware data cube for real-Time exploration of spatiotemporal data. *TVCG*, vol. 24, n° 3, p. 1394-1407.

Pahins C. A. L., Stephens S. A., Scheidegger C., Comba J. L. D. (2017). Hashedcubes: Simple, Low Memory, Real-Time Visual Exploration of Big Data. *TVCG*, vol. 23, n° 1, p. 671-680.

Silva R., Moura-Pires J., Santos M. Y. (2012). Spatial Clustering in SOLAP Systems to Enhance Map Visualization. *IJDWM*, , vol. 8, n° 2, p. 23-43.

Wang Z., Ferreira N., Wei Y., Bhaskar A. S., Scheidegger C. (2017). Gaussian Cubes: Real-Time Modeling for Visual Exploration of Large Multidimensional Datasets. *TVCG*, vol. 23, n° 1, p. 681-690.