# C-CUBE: Un nouvel opérateur d'agrégation pour les entrepôts de données en colonnes

**Khaled Dehdouh — Fadila Bentayeb — Nadia Kabachi — Omar Boussaid**

*Laboratoire ERIC, Université de Lyon 2*

*5 avenue Pierre Mendes-France, 69676 Bron Cedex, France*

RÉSUMÉ. *Les bases de données orientées colonnes offrent au domaine décisionnel le modèle le plus approprié au stockage des entrepôts de données. Cependant, en l'absence d'opérateurs d'analyse en ligne, le seul moyen, très coûteux, qui existe pour construire des cubes OLAP consiste à utiliser l'opérateur UNION sur des requêtes de regroupement afin d'obtenir l'ensemble des Group By nécessaires au calcul de cube OLAP*[1]*. Pour pallier ce problème, nous proposons dans cet article un nouvel opérateur d'agrégation, baptisé C-CUBE (Columnar-CUBE), qui permet de calculer des cubes de données à partir d'entrepôts de données stockés en colonnes. Nous avons implémenté l'opérateur C-CUBE au sein du SGBD orienté colonnes MonetDB et réalisé des expérimentations sur le benchmark SSBM* [2] *(Star Schema Benchmark). Nous avons ainsi pu montrer que C-CUBE présente des temps de calcul de cubes OLAP jusqu'à 70% moins élevés comparés à l'opérateur CUBE d'Oracle sur un entrepôt de 1To.*

ABSTRACT. *Columnar databases are suitable for data warehouses and multidimensional data structures storage. However, Columnar DBMS have not an appropriate operator for calculating OLAP cubes. In this paper, we propose a new OLAP operator for columnar DBMS, C-CUBE, that allows to calculate OLAP data cubes from columnar oriented-data warehouses. We have then implemented C-CUBE under MonetDB DBMS and curried out some experimentations onto Star Schema Benchmark. The obtained results show that C-CUBE improve the computation time of data cubes up to 70% compared to Oracle CUBE operator.*

MOTS-CLÉS : *Base de données orientée colonnes, Entrepôt de données, Cube OLAP.*

KEYWORDS: *Columnar databases, Data warehouse, OLAP Cube.*

1. On-Line Analytical Processing
2. SSBM: http://www.cs.umb.edu/poneil/StarSchemaB.PDF.

## 1. Introduction

Une base de donnée orientée colonnes stocke les données d'une table colonne par colonne. Cette technique de stockage permet d'avoir dans un même espace disque les valeurs appartenant à une même colonne, ce qui accélère énormément le temps d'accès à une colonne. Caractérisé par une opération de jointure très performante appelée jointure invisible (Abadi et al., 2008), le stockage en colonnes apparaît comme une solution très intéressante en BI (Business Intelligence) et plus particulièrement pour les entrepôts de données (Matei, 2010). Ces derniers sont dédiés à l'analyse en ligne pour l'aide à la prise de décision (Inmon, 1992). Grâce aux opérateurs OLAP (On- Line Analytical Processing), l'utilisateur peut extraire des cubes de données correspondants à des contextes d'analyse (Han et Kamber, 2006). Le stockage en colonnes est naturellement approprié à la structure de données multidimensionnelles et aux calculs d'agrégats (Stonebraker et al., 2005). Cependant, les fonctionnalités des bases de données orientées colonnes sont limitées. En effet, les SGBD (*Systèmes de Gestion de Bases de Données*) orientés colonnes ne disposent pas d'opérateurs de calcul de cubes OLAP. Toutefois, le cube OLAP peut être calculé avec la méthode naïve en utilisant l'union de requêtes de regroupement (*Group By*) (Dehdouh et al., 2013). Cette méthode de calcul engendre, pour un nombre de dimensions D, l'exécution de 2D sous-requêtes pour calculer les différents agrégats, ce qui augmente considérablement le nombre d'accès à la base de données. Par conséquent, la méthode naïve dégrade les performances du SGBD notamment pour le passage à l'échelle.

L'objectif de cet article est de proposer un opérateur d'agrégation C-CUBE (Columnar-CUBE) pour les SGBD orientés colonnes. Cet opérateur permet de calculer des cubes OLAP à partir d'entrepôts implémentés en colonnes. C-CUBE étend la jointure invisible, utilisée dans les bases de données orientées colonnes, pour prendre en considération toutes les combinaisons de dimensions. De plus, contrairement à la méthode naïve qui sollicite la base de données pour calculer les différents agrégats, C-CUBE exploite une vue (résultat d'une requête d'extraction) définie sur les attributs (dimensions et mesures) nécessaires au calcul du cube OLAP. Cette stratégie lui permet d'éviter le retour aux données de l'entrepôt pour le calcul d'agrégats. Nous avons validé cette technique sur un banc d'essais SSBM (Star Schema Benchmark) (O'Neil et al., 2009), que nous avons stocké au sein du SGBD orienté colonnes MonetDB, et nous avons mené ensuite des expérimentations qui ont permis de montrer, que C-CUBE optimise considérablement le temps de calcul des cubes OLAP comparé à la fois à la méthode naïve et à l'opérateur CUBE d'oracle.

La suite de cet article est organisée de la manière suivante. La section 2 dresse un état de l'art sur les bases de données orientées colonnes. La section 3 présente les différentes notions liées à notre travail. La section 4 est consacrée à la

description détaillée du processus de calcul de cube OLAP et à la présentation de l'opérateur C- CUBE. Nous avons procédé dans la section 5 à l'implémentation de notre opérateur C-CUBE et aux expérimentations d'évaluation. Enfin, nous concluons cet article et présentons quelques perspectives de notre travail dans la section 6.

## 2. État de l'art

Le stockage des données en colonnes remonte aux années 1970 avec l'utilisation des fichiers transposés et la technique de partitionnement vertical (D.S, 1979). C'est dans les années 1980 que les avantages de la décomposition des modèles de stockage ont été abordés dans la littérature. Ces modèles sont "DSM" (*Decomposition Storage Model*) et NSM (*N-ary Storage Model*) qui sont respectivement les prédécesseurs du stockage orienté colonnes et du stockage orienté lignes (Copeland et Khoshafian, 1985). Ce n'est que dans les années 2000 que le stockage des données en colonnes est apparu comme une alternative au stockage orienté lignes adopté par les SGBD relationnels notamment pour le stockage des bases de données multidimensionnelles (Abadi et al., 2008). Les travaux qui ont été menés dans ce domaine ont permis aux bases de données orientées colonnes de bénéficier indéniablement de meilleures performances relatives à l'opération de jointure, appelée jointure invisible. Rappelons que la jointure invisible est une opération de jointure qui utilise les positions des valeurs et les tables de hachage pour extraire les données qui satisfont les prédicats de la requête (Abadi et al., 2008). En outre, grâce à la technique de la matérialisation tardive (Abadi et al., 2007) et à la technique de compression des données qui agit efficacement sur des valeurs de même type, l'espace nécessaire pour le stockage des données a diminué considérablement (Abadi et al., 2006). Parmi les SGBD orientés colonnes, on peut citer C-Store[3], MonetDB[4] et Vertica[5]. Cependant, même si tout le monde s'accorde à dire que le stockage en colonnes est bien adapté aux données multidimensionnelles et par conséquent au calcul de cubes de données, les SGBD en colonnes ne disposent malheureusement pas d'opérateurs OLAP.

## 3. Calcul de cubes OLAP dans les entrepôts de données orientés colonnes

Dans cette section, nous définissons le cube OLAP et la méthode naïve qui permet de le calculer.

3. http ://db.csail.mit.edu/projects/cstore/
4. www.monetdb.org
5. http ://www.vertica.org/

### 3.1. *Cube OLAP*

Un cube de données est une collection de données agrégées et consolidées pour résumer l'information et expliquer la pertinence d'une observation. Il permet de représenter le fait à observer selon plusieurs axes d'observation (dimensions) (Gray et al.,: 1997) qui sont qualifiées de multidimensionnelles, indépendamment de leur support (tables relationnelles ou tableaux multidimensionnels). Le cube de données est exploré à l'aide de nombreuses opérations qui permettent sa manipulation (Rafanelli, 2003). Le calcul de cube permet d'avoir des agrégations au-delà des limites du Group by. Il calcule de façon multidimensionnelle et renvoie dans le cas de calcul d'une somme, des sous-totaux et totaux de toutes les combinaisons possibles. Cela consiste à calculer tous les agrégats suivant tous les niveaux des hiérarchies de toutes les dimensions. Pour un cube à trois dimensions A, B et C, les agrégats calculés concernent les combinaisons sui- vantes : (A, B, C), (A, B, ALL), (A, ALL, C), (ALL, B, C), (A, ALL, ALL), (ALL, B, ALL), (ALL, ALL, C), (ALL, ALL, ALL).

### 3.2. *La méthode naïve pour le calcul de Cube OLAP*

La méthode naïve pour calculer le cube de données est proposée par (Gray et al., 1997) et consiste à regrouper à l'aide de l'opérateur UNION, une collection de requêtes agrégatives exécutées séparément avec des Group By. Cependant, cette méthode présente l'inconvénient de solliciter de multiples accès à la base de données pour calculer les différents agrégats. En effet, pour un nombre de dimensions D, il y aura $2^D$ requêtes de regroupement à exécuter. Ce qui dégrade d'avantage les performances du système de gestion de base de données. De ce qui précède, il apparait clairement que la méthode naïve de calcul de cube OLAP n'est pas adaptée aux bases de données volumineuses, car c'est une approche caractérisée par une complexité qui est exponentielle par rapport au nombre de dimensions.

Pour pallier à ce problème et sachant que les SGBD en colonnes ne disposent pas d'opérateurs OLAP, nous proposons un nouvel opérateur d'agrégation, C-CUBE (Columnar-CUBE) permettant de calculer des cubes OLAP à partir d'entrepôts de données stockés en colonnes.

## 4. C-CUBE : Opérateur CUBE pour les bases de données en colonnes

Nous présentons dans cette section C-CUBE, notre opérateur OLAP pour le calcul de cube dans les entrepôts de données en colonnes. La technique utilisée par l'opérateur C-CUBE que nous proposons consiste à extraire à partir de l'entrepôt, les données qui satisfont tous les prédicats de la requête. Ces données sont regroupées en fonction des colonnes qui représentent les axes d'analyse. Le résultat est donc, une relation R composée des colonnes qui représentent les dimensions

(axes d'analyse) et de (des) colonne(s) représentant la (les) mesure(s) à agréger. La relation R est un résultat intermédiaire qui va servir à calculer l'ensemble des agrégats du cube OLAP. A ce stade, le résultat intermédiaire permet déjà d'obtenir l'agrégation totale et celle en fonction de toutes les colonnes représentant les dimensions. Pour obtenir les autres agrégations des sous-totaux qui composent le cube, chaque dimension au niveau du résultat intermédiaire est hachée avec les valeurs qui la composent pour obtenir des listes de positions, les valeurs de ces listes sont binaires, ils peuvent correspondre à "1" ou à "0", le "1" indique que la valeur de hachage existe à cette position et "0" si non. L'intersection avec un "*ET logique*" des listes de positions des différentes colonnes (dimensions) permet d'avoir un ensemble de listes de positions qui représentent les positions des valeurs de dimensions à combiner et les valeurs de la colonne mesure à agréger et ce, à différent niveau de granularité. Le regroupement de l'ensemble de ces résultats constitue le cube.

Pour détailler les phases d'exécution de cette technique avec un exemple, nous présentons dans la section suivante le banc d'essais décisionnel SSBM

### 4.1. *Banc d'essais décisionnel SSBM*

Pour décrire et illustrer nos différents propos et contributions, nous présentons dans cette section, le modèle de données en étoile SSBM (Star Schema Benchmark) (O'Neil et al., 2009) présenté dans la figure 1. Nous avons ensuite utilisé cet entrepôt pour évaluer les performances de l'opérateur C-CUBE.

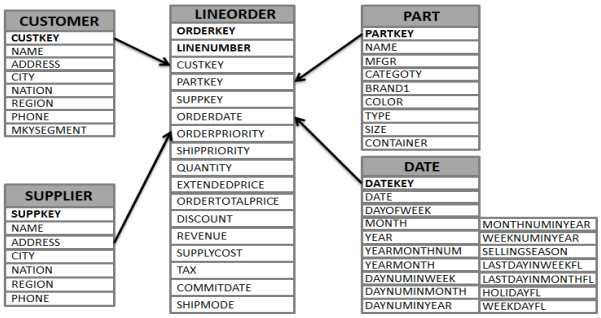


**Figure 1.** *Modèle en étoile de l'entrepôt de données SSBM*

SSBM est un banc d'essais décisionnel, dérivé du modèle TPC-H[6]. Contrairement à TPC-H, SSBM utilise le schéma en étoile pour permettre d'évaluer les performances de l'entrepôt de données. SSBM est un entrepôt de données qui

6. http:www.tpc.org/tpch/

gère les lignes de commandes en fonction des dimensions PART (produit), SUPPLIER (fournisseur), CUSTOMER (client) et DATE (date). Il est constitué d'une table de faits appelée LINEORDER (lignes de commandes) composée de dix-sept attributs pour renseigner une commande, dont la clé primaire est composée de ORDERKEY et de LINENUMBER et des clés étrangères provenant des tables de dimension. Ce modèle est accompagné d'une charge de requêtes avec de simple *Group by*.

Il est à noter que notre objectif n'est pas d'évaluer SSBM en tant que tel mais de l'utiliser pour évaluer la performance de notre opérateur C-CUBE dans un SGBD orienté colonnes. Pour cela, nous nous abstenons d'utiliser la charge de requêtes qui accompagne SSBM et nous créons de nouvelles requêtes permettant de calculer des cubes OLAP.

**Charge de requêtes :** Pour évaluer les performances de l'opérateur C-CUBE que nous proposons pour calculer le cube OLAP à partir d'entrepôt stocké en colonnes, nous avons utilisé quatre requêtes de calcul des cubes OLAP. Ces requêtes impliquent graduellement le nombre de dimensions dans le calcul des cubes.

– Requête 1 : C'est une requête qui permet de calculer un cube OLAP à deux dimensions avec des restrictions sur les colonnes, NATION de la dimension CUSTOMER, NATION de la dimension SUPPLIER et YEAR de la dimension DATE. Cette requête calcule à différents niveaux de granularité, la somme des revenus (*sum(revenue)*) en fonction des attributs, CITY de la dimension CUSTOMER et CITY de la dimension SUPPLIER.

– Requête 2 : C'est une requête qui permet de calculer un cube OLAP à trois dimensions avec les mêmes restrictions de la première requête. Elle calcule à différents niveaux de granularité, la somme des revenus (*sum (revenue)*) en fonction des attributs CITY de la dimension CUSTOMER, CITY de la dimension SUPPLIER et YEAR de la dimension DATE.

– Requête 3 : C'est une requête qui permet de calculer un cube OLAP à quatre dimensions avec des restrictions sur les colonnes NATION de la dimension CUSTOMER, NATION de la dimension SUPPLIER, BRAND1 de la dimension PART et YEAR de la dimension DATE. Elle calcule à différents niveaux de granularité, la somme des revenus (*sum (revenue)*) en fonction des attributs CITY de la dimension CUSTOMER, CITY de la dimension SUPPLIER, YEAR de la dimension DATE et BRAND1 de la dimension PART.

– Requête 4 : C'est une requête qui permet de calculer un cube OLAP à cinq dimensions avec les mêmes restrictions de la requête précédente (requête 3). Elle calcule à différents niveaux de granularité, la somme des revenus (*sum (revenue)*) en fonction des attributs CITY de la dimension CUSTOMER, CITY de la dimension SUPPLIER, YEAR et MONTH de la dimension DATE et BRAND1 de la dimension PART.

### 4.2. *Phases d'exécution de l'opérateur C-CUBE*

L'opérateur C-CUBE calcule le cube OLAP en quatre phases. Pour détailler ces phases, nous illustrons notre explication avec un exemple de la requête 2, citée dans la section 4.1. L'exemple calcule la somme des revenus des ventes des produits livrés par des fournisseurs algériens et commandés par des clients français pendant les années 1996 et 1997.

**Première phase :** Elle consiste à extraire, à partir de l'entrepôt de données, les données qui satisfont tous les prédicats (filtres). Cette phase permet d'obtenir le résultat intermédiaire pour constituer l'ensemble des parties du cube. Pour réaliser cette phase, les prédicats de la requête sont appliqués séparément sur les dimensions respectives pour obtenir les listes des clés primaires des dimensions qui satisfont les prédicats. Vu que ces clés sont des clés étrangères au niveau de la table de faits, elles sont utilisées pour extraire les listes des positions y afférant au niveau de la table de faits. L'association de ces listes de positions avec un "*ET logique*" génère une seule liste de positions P. Cette dernière représente les positions des valeurs dans la table de faits qui satisfont tous les prédicats de la requête à la fois. L'extraction des valeurs de la table de faits en fonction de P permet d'obtenir le résultat intermédiaire R sous forme d'une vue qui va servir à calculer le cube OLAP. Dans notre exemple, cela revient à sélectionner les clés primaires des villes algériennes (Nation = Algeria) de la dimension SUPPLIER, celles des villes françaises de la dimension CUSTOMER (Nation = France) et celles des années 1996 et 1997 de la dimension DATE (Year in (1996, 1997)). Cette opération donne lieu à trois listes de clés.

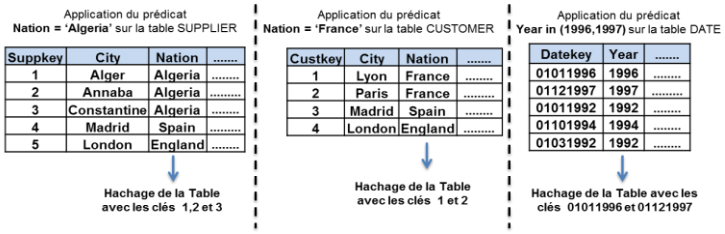


**Figure 2.** *Extraction des clés des dimensions qui satisfont les prédicats respectives*

Pour obtenir les listes de positions correspondantes à ces clés dans la table de faits, les dimensions Suppkey, Custkey et Orderdate de la table de faits LINEORDER sont hachées sur les trois listes des clés respectives. Cette opération donne lieu à trois listes de positions. Ces dernières, une fois associées génèrent, une liste de positions P qui représente les positions des n-uplets qui satisfont tous les prédicats de la requête. Cette opération donne lieu à un résultat intermédiaire appelé R. A ce stade de l'exécution, les agrégations des sommes des revenus de

(ALL, ALL, ALL) et (Suppkey, Custkey, Orderdate) sont calculées telles que c'est décrit dans la figure 3.

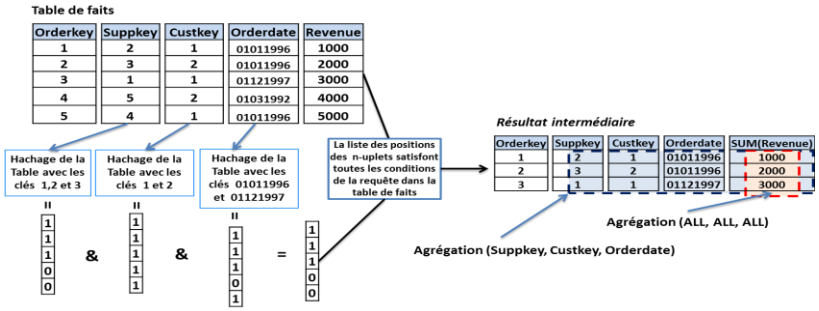


**Figure 3.** *Extraction des données qui satisfont tous les prédicats de la requête et construction du résultat intermédiaire (relation R)*

**Deuxième phase :** Dans cette phase, chaque colonne du résultat intermédiaire R représentant une dimension est hachée avec les valeurs qui la composent pour obtenir les listes des positions de ces valeurs. En effet, la dimension Suppkey du résultat intermédiaire est hachée sur les valeurs (2, 3 et 1) donnant lieu respectivement à trois listes de positions $P_{Suppkey(2)}$, $P_{Suppkey(3)}$ et $P_{Suppkey(1)}$. Custkey est hachée sur les valeurs (1 et 2) donnant lieu à $P_{Custkey(1)}$ et $P_{Custkey(2)}$ et enfin, Orderdate est hachée sur (01011996 et 01121997) donnant lieu à $P_{Orderdate(1996)}$ et $P_{Orderdate(1997)}$. A ce stade de l'exécution, ces listes de positions permettent d'agréger les valeurs de la colonne représentant la mesure, et ce, pour chaque dimension séparément. En effet, elles fournissent les agrégations des sommes des revenus suivantes : (Suppkey, ALL, ALL), (ALL, Custkey, ALL) et (ALL, ALL, Orderdate).

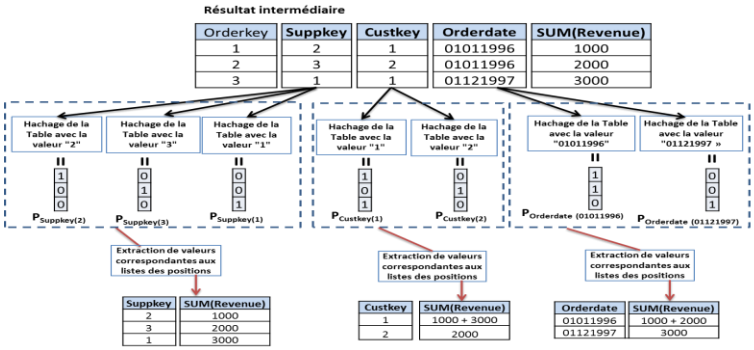


**Figure 4.** *Construction des listes de positions et calcul d'agrégats pour chaque dimension du résultat intermédiaire.*

**Troisième phase :** Au niveau de cette phase, les listes de positions des dimensions au niveau de R sont associées via l'opérateur "*ET logique*". Cette opération permet d'identifier les valeurs des dimensions à combiner et les valeurs de la mesure à agréger qui correspondent aux différentes combinaisons. Dans ce cas, pour identifier les différentes combinaisons possibles entre les dimensions Suppkey et Custkey, les listes de positions $P_{Suppkey(2)}$, $P_{Suppkey(3)}$ et $P_{Suppkey(1)}$ sont associées avec $P_{Custkey(1)}$ et $P_{Custkey(2)}$ via l'opérateur "*ET logique*". Les résultats sont les trois listes $P_{Suppkey(2)-Custkey(1)}$, $P_{Suppkey(3)-Custkey(2)}$ et $P_{Suppkey(1)-Custkey(1)}$. Ces listes permettent d'extraire les valeurs des dimensions Suppkey, Custkey et de mesure Revenue à agréger. Ces opérations fournissent les agrégations des sommes des revenus suivantes : (Suppkey, Custkey, ALL), (Suppkey, ALL, Orderdate) et (ALL, Custkey, Orderdate).

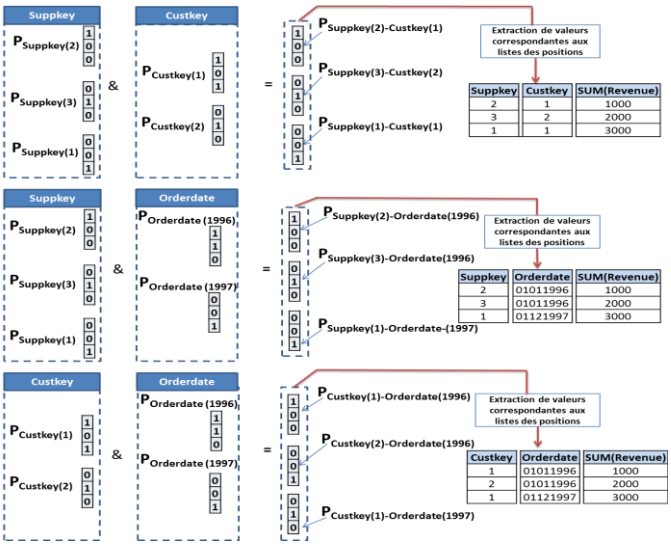


**Figure 5.** *Calcul d'agrégat pour les différentes combinaisons des listes de positions*

**Quatrième phase :** Elle consiste à regrouper toutes les combinaisons et les agrégats réalisés. Ensuite, elle extrait les valeurs à afficher correspondantes aux clés des dimensions. En effet, la construction de toutes les combinaisons pour calculer les différents agrégats qui constituent le cube OLAP a été réalisée avec les clés de dimensions. Dans le cas de notre exemple, les clés (2, 3, 1) de la dimension Suppkey correspondent aux valeurs (Annaba, Constantine, Alger) de la dimension SUPPLIER, les clés (1, 2) de la dimension Custkey correspondent aux valeurs (Lyon, Paris) de la dimension CUSTOMER et enfin les clés (01011996, 01121997) correspondent aux valeurs (1996, 1997) de la dimension DATE. Par conséquent,

le regroupement des sous-résultats (totaux et sous-totaux) des trois phases précédentes permet de calculer le cube OLAP représenté dans la figure 6.

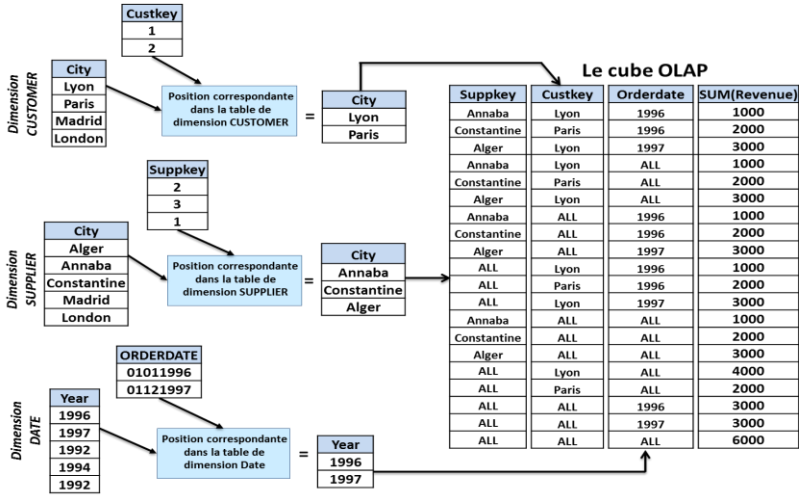


**Figure 6**. *Regroupement des agrégats (totaux et sous-totaux) et extraction des valeurs à afficher, correspondantes aux clés de dimensions.*

Noter bien que la quasi-totalité des traitements ont été effectués avec des clés et des listes de positions. Ce procédé est très avantageux aux traitements au niveau de la mémoire car il offre la possibilité de réaliser le maximum des opérations, sans solliciter pour autant les accès au disque, ce qui permet de réduire considérablement le flux d'entrées/sorties.

## 5. Implémentation et expérimentations

### 5.1. *Implémentation*

Pour valider notre méthode de calcul de cube OLAP à partir d'entrepôt de données orienté colonnes, nous avons implémenté l'opérateur C-CUBE en JAVA. Par ailleurs, pour mener nous expérimentations, nous avons implémenté le banc d'essais décisionnel SSBM, décrit dans la section 4.1, sous le SGBD MonetDB (Idreos et al., 2012). Le choix de ce dernier est motivé par le fait qu'il est orienté colonnes et en sources libres (open sources). L'environnement de tests que nous avons utilisé consiste en une machine intel-Core TMi3-3220 CPU@3.30 GHZ avec une mémoire RAM de 4Go. Cette machine fonctionne avec le système d'exploitation Microsoft Windows 7 de 64bits.

### 5.2. *Expérimentations*

Dans cette partie de l'article, nous avons évalué les performances de l'opérateur C-CUBE en matière de temps de calcul du cube OLAP sur l'entrepôt de données SSBM selon deux architectures. La première est relationnelle orientée lignes avec le SGBD Oracle 11g. La deuxième est relationnelle orientée colonnes avec le SGBD MonetDB. Les deux implantations nous permettent de comparer le temps d'exécution des requêtes de construction des cubes OLAP selon l'opérateur C-CUBE, la méthode naïve et l'opérateur CUBE d'oracle. Nous avons ensuite mené deux expérimentations. La première évalue le temps de calcul des cubes OLAP avec un nombre de dimensions qui augmente graduellement. La deuxième évalue le gain en matière de temps de calcul de cube OLAP entre l'opérateur C-CUBE et l'opérateur CUBE en faisant varier la taille de l'entrepôt.

#### 5.2.1. *Calcul des cubes OLAP*

L'objectif de cette expérimentation est d'observer le comportement de notre opérateur C-CUBE face aux variations du nombre de dimensions. Nous avons fixé la taille de l'entrepôt de données à 50 Go. Ensuite, nous avons comparé la performance de C-CUBE avec la méthode naïve de l'approche orientée colonnes et l'opérateur CUBE du SGBD Oracle. Pour cela, nous avons exécuté les quatre requêtes présentées dans la section 4.1. Ces requêtes calculent des cubes OLAP avec un nombre de dimensions qui augmente progressivement. Les résultats que nous avons obtenus sont présentés dans la figure 7.

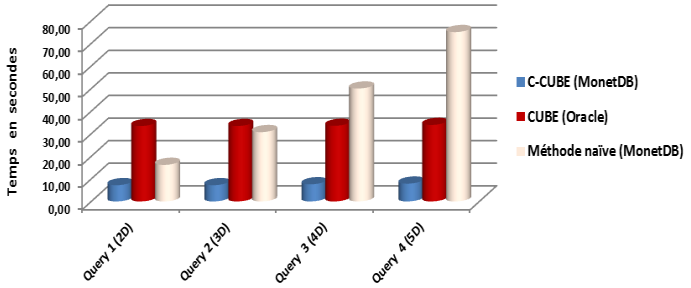


**Figure 7.** *Résultats de calcul des cubes OLAP à 2, 3, 4 et 5 dimensions*

Nous constatons que les temps de calcul des cubes OLAP avec la méthode naïve varient en moyenne entre 16.2 et 75 secondes. Ces temps de calcul augmentent en fonction du nombre de dimensions. En effet, le temps nécessaire pour calculer un cube à deux dimensions est de 16.2, et pour un cube à trois dimensions est de 30.7 secondes, cela représente presque le double. Au-delà de trois dimensions, cette méthode enregistre de mauvais résultats par rapport aux autres méthodes. Ce

résultat est expliqué par le fait que pour deux dimensions, le système exécute quatre ($2^2$) requêtes agrégatives, souvent avec des jointures répétitives et des regroupements de résultats. Par contre, avec trois dimensions, il exécute huit ($2^3$) requêtes agrégatives, ce qui représente pratiquement le double. Cependant, l'augmentation du nombre de dimension implique des accès disques importants pour extraire les données de l'entrepôt, cela engendre au niveau de la mémoire une gestion supplémentaire des résultats intermédiaires. Par conséquent, il en découle une saturation de la mémoire, cette dernière sollicite le disque pour gérer les résultats intermédiaires. De ce fait, il est clair que ce procédé augmente considérablement le coût d'entrées/sorties qui se traduit par un temps d'exécution plus élevé et une dégradation des performances du système.

En revanche, nous constatons une légère variation des temps d'exécution des requêtes générant les cubes OLAP, avec les opérateurs C-CUBE et CUBE. Cependant, C-CUBE affiche une meilleure performance que CUBE, en effet, l'opérateur CUBE enregistre des temps entre 33.4 et 33.9 secondes, alors que l'opérateur C-CUBE enregistre des temps entre 7.2 et 7.9 secondes.

L'avantage de C-CUBE réside dans l'utilisation et l'exploitation des listes de positions pour le calcul des cubes OLAP. Ces listes occupent peu d'espace mémoire, elles conviennent parfaitement à un traitement au niveau de la mémoire sans pour autant solliciter de multiples accès au disque. En effet, l'augmentation du nombre de dimensions dans le calcul de cube de données se traduit techniquement par la création et la manipulation des vecteurs de bits qui représentent des listes des positions des valeurs. Ce procédé n'impacte pas lourdement la mémoire. De plus, les combinaisons sont réalisées avec des listes (vecteur de bits composé de "1" ou "0") et non pas avec des valeurs, ces dernières ne sont extraites qu'après avoir construit la liste des positions y afférente à une combinaison.

Eu égard des résultats obtenus, nous avons constaté que l'opérateur C-CUBE que nous avons proposé optimise considérablement le temps de calcul de cube OLAP. Pour cela, il était très intéressant d'évaluer le comportement face à un passage à l'échelle en augmentant la taille de l'entrepôt jusqu'à 1 To.

### **5.2.2.** *Calcul des cubes OLAP avec un passage à l'échelle*

L'objectif de cette expérimentation est d'évaluer en pourcentage le gain en matière de temps de calcul de cube OLAP offert par l'opérateur C-CUBE, par rapport à l'opérateur CUBE des SGBDR orientés lignes. Pour cela, nous avons confronté l'opérateur C-CUBE au passage à l'échelle. L'expérimentation consiste à évaluer le temps d'exécution de la requête 2 de la section 4.1 qui calcule un cube OLAP à trois dimensions, avec des échantillons de données qui augmentent progressivement, allant de 100 G0 jusqu'à 1 To. Les résultats que nous avons obtenus sont présentés dans la figure 8.

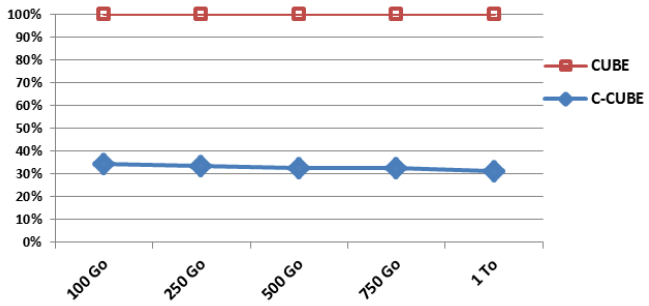


**Figure 8.** *Résultat de calcul des cubes OLAP avec un passage à l'échelle*

Cette figure représente la tendance en pourcentage des temps de calcul des cubes OLAP entre les opérateurs C-CUBE et CUBE dans des différentes tailles de l'entrepôt de données.

Nous constatons que la courbe représentant le temps de calcul des cubes OLAP avec l'opérateur C-CUBE présente de meilleurs résultats et ce, quelle que soit la taille de l'entrepôt. En effet, L'écart des résultats des temps de calcul des cubes OLAP des deux opérateurs varie en générale entre 65% et 70% pour une taille de l'entrepôt allant de 100 Go à 1 To. Cet écart en faveur de l'opérateur C-CUBE démontre clairement son avantage.

La performance de l'opérateur C-CUBE s'explique par le fait que le système exploite une vue (résultat d'une requête d'extraction) définie sur les attributs (dimensions et mesures) nécessaires au calcul de cube OLAP. Cette stratégie lui permet de réaliser les différentes combinaisons et le calcul d'agrégats au niveau de la mémoire et d'éviter le retour aux données de l'entrepôt. En effet, Cela est possible grâce à l'utilisation de la position de la valeur au lieu de la valeur elle-même. Les positions de valeurs représentées par les listes de vecteurs n'occupent pas beaucoup d'espace au niveau de la mémoire. Ce qui offre la possibilité d'effectuer plusieurs traitements au niveau de la mémoire et diminue par conséquent considérablement le flux d'entrée et sortie.

Au final, les expérimentations que nous avons réalisées démontrent clairement que l'opérateur C-CUBE que nous avons proposé pour calculer le cube OLAP dans les bases de données orientées colonnes est performant par rapport à l'opérateur CUBE d'Oracle.

## 6. Conclusion

L'intérêt majeur de ce travail est d'étendre l'utilisation des bases de données en colonnes au domaine décisionnel. Dans ce contexte, nous avons proposé dans cet article un opérateur de calcul du cube OLAP, baptisé C-CUBE. L'avantage de cet opérateur est qu'il s'appuie sur le principe de la jointure invisible qui est exploitée ici pour calculer de façon efficace les agrégats du cube OLAP. En effet, à l'instar de la technique utilisée dans la jointure invisible, C-CUBE manipule des listes des positions des valeurs et des tables de hachage qui contiennent des clés des différentes dimensions invoquées dans la requête. Cette manière de procéder réduit considérablement le flux d'entrée et sortie. L'implémentation de cet opérateur au sein du SGBD orienté colonnes MonetDB, et les expérimentations que nous avons menées sur l'entrepôt de données relationnel SSBM ont montré clairement la performance de notre opérateur comparée à celle de l'opérateur CUBE d'Oracle.

De manière plus générale, l'opérateur C-CUBE peut être appliqué sur tout SGBD orienté colonnes et peut être généralisé même aux SGBD orientées colonnes non relationnels (NoSQL[7]). C'est dans ce contexte que ce travail ouvre plusieurs perspectives de recherche intéressantes. L'une des pistes de recherche consiste à adapter l'opérateur C-CUBE aux calculs des cubes OLAP à partir des entrepôts de données NoSQL qui gèrent des Big Data à grande échelle (Jerzy, 2012).

## 7. Bibliographies


Abadi D., Madden S., Ferreira M., « Integrating compression and execution in column oriented database systems », *Special Interest Group on Management of Data Conference*, 2006, p. 671-682.

Abadi D., Madden S., Hachem N., « Column-stores vs. row-stores: how different are they really? », *International Conference on Management of Data*, p2008, 967-980.

Copeland G., Khoshafian S., « A decomposition storage model », *Special Interest Group on Management Of Data Record*, 1985, p. 268-279.

Dehdouh K., Bentayeb F., Kabachi N., « Performances de requêtes OLAP dans les bases de données en colonnes », *Conférence sur les Avancées des Systèmes Décisionnels*, 2013, p. 439-444.

Batory, D., « On searching transposed files », *Association for Computing Machinery*, 1979, p.531-544.

Gray J., Chaudhuri S., Bosworth A., Layman A., Reichart D., Venkatrao M., Pellow F., Pirahesh H., « Data cube : A relational aggregation operator generalizing group-by,


7. Not Only Sql

cross-tab, and sub totals », *Journal of Data Mining and Knowledge Discovery*, 1997, p.29-53.

Han J., Kamber M., Data mining : concepts and techniques, Morgan Kaufmann Publishers Inc, San Francisco, CA, USA, 2006.

Idreos v., Groffen F., Nes v., Manegold S., Mullender K., Sjoerd K., Kersten v., « MonetDB : Two Decades of Research in Column-oriented Database Architectures », *Journal IEEE Data Engineering. Bull*, 2012, p.40-45.

Inmon W., « Building the Data Warehouse », 1992.

Jerzy D., « Business Intelligence and NoSQL Databases », *Information Systems in Management*, 2012, p.25-37.

Matei G., « Column-Oriented Databases, an Alternative for Analytical Environment », *Database Systems Journal*, 2010, p. 3-16.

O'Neil P., O'Neil B., Chen X., « The Star Schema Benchmark (SSBM) », 2009. http ://www.cs.umb.edu/p˜oneil/StarSchemaB.PDF.

Rafanelli M., « Operators for Multidimensional Aggregate Data », *Multidimensional Databases: Problems and Solutions, IGI Publishing Group*, 2003, p. 116-165.

Stonebraker v., Abadi D., Batkin A., Chen X., Cherniack M., Ferreira M., Lau E., Lin v., Madden S., O'Neil E., O'Neil P., Rasin A., Tran N., Zdonik S., « C-store : a column-oriented DBMS », *Proceedings of the 31st International Conference on Very Large Data Bases*, 2005, p. 553-564.