IMPLANTATION NOT ONLY SQL DES BASES DE DONNEES MULTIDIMENSIONNELLES

Max Chevalier (*), Mohammed El Malki (*,), Arlind Kopliku (*), Olivier Teste (*), Ronan Tournier (*)**
{Max.Chevalier, Mohammed.ElMalki, Arlind.Kopliku, Olivier.Teste, Ronan.Tournier}@irit.fr

(*) IRIT (UMR 5505), Université de Toulouse, 118 route de Narbonne, 31062, Toulouse Cedex 9, France,
(**) Capgemini, 109 avenue du Général Eisenhower, BP 53655- 31036 Toulouse, France.

Mots clefs :
Conception d'entrepôts de données, modélisation multidimensionnelle, bases de données NoSQL orienté documents, transformation de modèles
Keywords:
Data warehouse design, multidimensional modeling, document-oriented NoSQL databases, model transformation

Résumé

Les systèmes NoSQL (Not Only SQL) se développent notamment grâce à leurs capacités à gérer facilement de grands volumes de données, et leur flexibilité en terme de type de données. Dans cet article, nous étudions l'implantation d'un entrepôt de données multidimensionnelles avec un système NoSQL orientée documents. Nous proposons des règles de transformation qui permettent de passer d'un modèle conceptuel multidimensionnel vers un modèle logique NoSQL orienté documents. Nous proposons trois types de transformation pour implanter les entrepôts de données multidimensionnelles. Nous expérimentons ces trois approches avec le système MongoDB, et étudions le chargement des données, les processus de transformation d'un type d'implantation à un autre ainsi que le pré-calcul d'agrégats inhérents aux entrepôts de données multidimensionnelles.

1 Introduction

Les systèmes NoSQL démontrent certains avantages sur les systèmes de gestion de bases de données relationnelles [15]. De nos jours, ces systèmes sont utilisés pour le stockage des mégadonnées[1] (« Big Data ») et leur analyse. Ce travail étend nos précédents travaux sur l'usage de solutions NoSQL pour les entrepôts de données [2], et rejoint un certain nombre de travaux de recherche en cours [6], [9], [18]. Dans cet article nous considérons un seul type de système NoSQL, à savoir les systèmes orientés documents [7].

Les systèmes orientés documents sont l'une des familles de systèmes NoSQL les plus connues. Les données y sont stockées sous forme de collections de documents, un document correspondant à un enregistrement. Chaque document est constitué de paires clés/valeurs, une valeur pouvant être elle-même composée de paires clé/valeurs, à savoir un document imbriqué. Les bases de données orientées documents permettent une flexibilité accrue dans la conception du schéma car elles permettent le stockage de données aux structures complexes et hétérogènes dans une même collection. Bien que ces bases soient considérées sans schéma prédéfini (« schemaless »), la majorité des cas d'utilisation nécessite quand même un modèle de données.

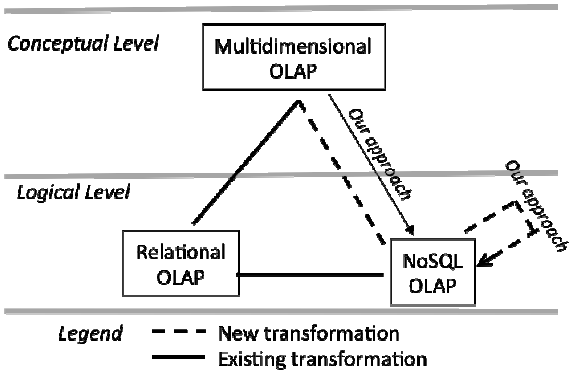


Figure 1 : Transformations d'un modèle multidimensionnel du conceptuel au logique.

Dans le cadre des entrepôts de données, des travaux existants ont montré qu'il était possible de les implanter avec différents modèles logiques [12]. Les entrepôts de données reposent principalement sur un modèle de données multidimensionnelles. Ce dernier est un modèle conceptuel[2] et il est nécessaire, afin de l'implanter, de le transformer en un modèle logique représentant une technologie ; dans notre cas, un modèle NoSQL orienté documents. La transformation d'un modèle conceptuel en un modèle relationnel, à savoir un modèle logique représentant la technologie des bases de données principalement utilisées dans les entrepôts de données, est directe. Toutefois à ce jour, aucun travail de recherche ne considère une implantation directe d'un modèle conceptuel multidimensionnel dans un modèle logique

[1] Mégadonnées : définition des données massives (« Big Data »), JORF du 22 08 2014.
[2] Un modèle au niveau conceptuel consiste à décrire les données d'une manière générique indépendamment des technologies de l'information alors qu'un modèle du niveau logique va utiliser une technologie spécifique pour implanter les concepts représentés par le modèle au niveau conceptuel. On parle de modèles conceptuels et logiques.

NoSQL (*cf*. Figure 1). Les modèles NoSQL supportent des structures de données plus complexes ; autrement dit, les structures supportent des attributs atomiques ainsi que des attributs complexes (documents imbriqués) pour décrire les données. Ils ont une structure flexible permettant de différentier les attributs constituant deux documents d'une même collection. Dans ce contexte, il est envisageable de proposer différentes structures logiques pour implanter un modèle conceptuel multidimensionnel en exploitant notamment le principe d'imbrication. En outre, l'évolution des besoins des utilisateurs (des décideurs) peut requérir de transformer des structures logiques existantes vers une autre organisation.

Cet article présente notre étude sur les processus de transformation modèle à modèle des entrepôts de données multidimensionnelles dans le contexte des systèmes NoSQL orientés documents. Nous comparons trois processus de transformation pour obtenir un modèle logique à partir d'un modèle conceptuel. Nous définissons un formalisme pour décrire précisément les règles de passage de modèles du niveau conceptuel vers le niveau logique. Notre étude inclut le chargement des données dans le système final, les conversions entre modèles ainsi que le pré-calcul des agrégats (parfois appelés « vues matérialisées » du cube OLAP — *On-Line Analytical Processing* ou processus d'analyse en ligne).

Nos motivations d'implanter un système OLAP en utilisant un système NoSQL comme nouvelle alternative est multiple [6],[15] :

[1] la capacité des systèmes NoSQL à gérer de très grands volumes de données ;
[2] la formalisation de ces environnements permettant notamment de distinguer clairement les éléments structurels des valeurs ;
[3] la recherche de modèles pouvant servir de modèle générique à défaut de disposer actuellement de notations standards; et
[4] une évaluation des différents systèmes NoSQL.

Cet article s'articule comme suit. La section 2 étudie l'état de l'art du domaine. En section 3, nous formalisons le modèle conceptuel multidimensionnel de données ainsi que les agrégats pré-calculables. Nous présentons ensuite la formalisation du modèle logique orienté documents, et les définitions des règles de transformation. En section 4 nous présentons et discutons les résultats de nos expérimentations.

2 État de l'art

Un nombre non négligeable de recherches se sont focalisées sur la transformation des concepts d'entrepôts de données multidimensionnels vers un environnement logique relationnel ; ces approches sont qualifiées de R-OLAP [1], [5]. Ces approches consistent à implanter les bases de données multidimensionnelles en utilisant des systèmes de gestion de bases de données relationnelles (SGBDR). Des règles de passage sont utilisées pour convertir les structures multidimensionnelles du niveau conceptuel (faits, dimensions et hiérarchies) en un modèle logique basé sur des relations. En outre, de nombreux travaux [10] ont étudié des méthodes d'optimisation basées sur des agrégats pré-calculés (vues matérialisées, OLAP cuboïdes…). Toutefois les implantations R-OLAP souffrent du passage à l'échelle pour les très grands volumes de données qui émergent aujourd'hui avec les mégadonnées ou *big data*. Des pistes sont explorées pour de nouvelles solutions utilisant NoSQL [15]. Notre approche vise à revisiter ces processus pour implanter automatiquement des modèles conceptuels multidimensionnels en modèles NoSQL.
D'autres études ont étudié le processus de transformation d'un schéma relationnel en un modèle logique NoSQL (*cf*. Figure 1). Dans [14], un algorithme est introduit pour faire correspondre un schéma relationnel en un schéma NoSQL MongoDB [7], un système orienté document. Néanmoins ces approches se limitent à des transformations d'une représentation logique vers une autre représentation logique (modèle relationnel vers modèle orienté documents) et ne prennent pas en considération le niveau conceptuel (et par conséquent toutes les contraintes décrites par les structures de ce niveau). En complément, dans [13], l'auteur propose une approche pour optimiser le schéma NoSQL.

A notre connaissance, aucune approche n'a été définie pour transformer directement et automatiquement un modèle conceptuel multidimensionnel d'entrepôt de données en un modèle logique NoSQL. Il est possible de transformer des modèles conceptuels multidimensionnels en modèles logiques ROLAP et, dans un second temps, de transformer ces modèles relationnels en modèles logiques NoSQL. Toutefois, cette transformation, utilisant le modèle relationnel comme modèle pivot intermédiaire, n'a pas été formalisée car les deux transformations ont été étudiées de manière indépendante. Nos précédents travaux [2] [4] considèrent deux modèles NoSQL (un orienté colonnes et un orienté documents), avec un processus de transformation unique. Dans cet article nous focalisons l'étude sur le modèle orienté documents afin de développer trois processus de structuration logique.

3 Modèle conceptuel multidimensionnel et cube OLAP

Dans cette section nous définissons le modèle conceptuel [15] [16] sur lequel repose notre proposition. Nous y ajoutons la formalisation d'un cube OLAP, qui représente l'ensemble des agrégats pré-calculés (parfois appelés OLAP cuboïdes).

3.1 Modèle conceptuel multidimensionnel

Nous introduisons un formalisme permettant de décrire les éléments multidimensionnels utilisés au niveau conceptuel. La spécification des transformations (*cf.* section 4) repose sur ces définitions conceptuelles.

Un **schéma multidimensionnel**, noté E, est défini par (F^E, D^E, $Star^E$) où :

- $F^E = \{F_1, ..., F_n\}$ est un ensemble fini de n faits,
- $D^E = \{D_1, ..., D_m\}$ est un ensemble fini de m dimensions,
- $Star^E$: $F^E \rightarrow 2^{D^E}$ est une fonction qui associe à chaque fait de F^E un ensemble de dimensions qui peuvent être utilisées pour analyser le fait (2^{D^E} représentant l'ensemble des parties de D^E).

Une **dimension**, noté $D_i \in D^E$ (notée de manière abusive D), est définie par (N^D, A^D, H^D) où :

- N^D est le nom de la dimension,
- $A^D = \{a_1^D, ..., a_u^D\} \cup \{id^D, All^D\}$ $A^D = \{a_1^D, ..., a_u^D\} \cup \{id^D, all^D\}$ est un ensemble de u+2 attributs de la dimension,
- $H^D = \{H_1^D, ..., H_h^D\}$ est un ensemble de h hiérarchies.

Une **hiérarchie** de la dimension D, $H_i \in H^D$, est définie par (N^{Hi}, $Param^{Hi}$, $Weak^{Hi}$) où :

- N^{Hi} est le nom de la hiérarchie,
- $Param^{Hi} = < id^D, p_1^{Hi}, ..., p_{V_i}^{Hi}, All^D >$ $Param^{H_i} = \langle id^D, p_1^{H_i}, ..., p_w^{H_i}, All^D \rangle$ est un ensemble ordonné de w+2 attributs ($w \leq u$) appelés **paramètres** fournissant une échelle graduée de la hiérarchie, $\forall k \in [1..w]$, p_k^{Hi} $p_k^{Hi} \in A^D$,
- $Weak^{Hi}$: $Param^{Hi} \rightarrow 2^{A^D - Param^{Hi}}$ $2^{A^D - param^{Hi}}$ est une fonction associant à chaque paramètre d'éventuelles informations complémentaires appelées **attributs faibles**.

Un **fait** $F \in F^E$, est défini par (N^F, M^F) où :

- N^F est le nom du fait,
- $M^F = \{f_1(m_1^F), \ldots, f_v(m_v^F)\}$ $M^F = \{f_1(m_1^F), \ldots, f_v(m_v^F)\}$ est un ensemble de **mesures**, chacune associée avec une **fonction d'agrégation** f_i.

3.2 Cube OLAP

Le **treillis d'agrégats pré-calculés** (aussi parfois appelé treillis de cuboïdes OLAP) correspond à un ensemble de cuboïdes, chacun étant un sous ensemble de mesures d'un fait, agrégées selon un sous-ensemble de dimensions connectées au fait. Un cuboïde correspond à une requête d'analyse (par exemple, analyser la quantité de produits vendus par catégorie de produit et par mois). Les cuboïdes OLAP sont pré-calculés et stockés à l'avance pour accélérer le temps de réponse aux requêtes évitant d'en calculer le résultat. Typiquement, les données des mesures sont regroupées selon un sous-ensemble de dimensions (ex. grouper par catégorie et par mois) et des fonctions d'agrégation sont utilisées pour agréger les données des mesures (somme des quantités vendues) selon les groupes créés. Formellement, un cuboïde O dérivé de E, $O=(F^O,D^O)$ est composé de :

- $F^O = (N^{Fo}, M^{Fo})$ est un fait dérivé de $F \in F^E$ avec $N^{Fo} = N^F$ et un sous-ensemble de ses mesures $M^{Fo} \subseteq M^F$.
- $D^O \subseteq Star^E(F^O) \subseteq D^E$ est un sous ensemble des dimensions D^E. Plus précisément, D^O est une combinaison des dimensions de $Star^E(F^O)$ associées à F^O.

Si nous générons les cuboïdes OLAP d'un fait selon toutes les combinaisons de toutes les dimensions qui lui sont associées, à savoir, $D^0 \subseteq 2^{Star^E(F)}$, nous obtenons un **treillis d'agrégats**, ou un **treillis de cuboïdes** appelé aussi **cube OLAP** [1], [10]. Il est à noter également qu'une « étoile », à savoir un fait ($F \in F^E$) et ses dimensions associées ($Star^E(F)$), peut également être représentée par un cuboïde où $F^O=F$ et $D^O=Star^E(F)$.

Exemple. Nous utilisons un extrait du jeu de données *Star Schema Benchmark*, SSB [3] [14]. Ce jeu de données consiste en une analyse de commandes de clients. Les commandes sont faites par des clients et sont constituées de lignes (*LineOrder*) qui sont analysées. Une ligne consiste en un produit (*Part*) acheté auprès d'un fournisseur (*Supplier*) et vendu à un client (*Customer*) à une date donnée (*Date*). Les commandes sont analysées par la quantité vendue (*Quantity*), les réductions appliquées (*Discount*), le bénéfice (*Revenue*) et le montant des taxes (*Tax*).

Le schéma conceptuel multidimensionnel est représenté en Figure 2 en utilisant des notations graphiques issues de [9] [16]. Ainsi le schéma est constitué d'un fait $F^{LineOrder}$ qui est défini par (*LineOrder*, {*SUM*(*Quantity*), *SUM*(*Discount*), *SUM*(*Revenue*), *SUM*(*Tax*)}). Ce fait est analysé selon quatre dimensions, chacune étant constituée de plusieurs paramètres (niveaux de détails) organisés hiérarchiquement :

- La dimension client ($D^{Customer}$) avec les paramètres code du client (*Customer*) — accompagné du nom du client en attribut faible (*Name*) —ville (*City*), région (*Region*) et pays (*Nation*) ;
- La dimension produit (D^{Part}) avec les paramètres code produit (Partkey) — accompagné du nom du produit et de sa taille en attributs faibles (*Prod_Name* et *Size*) — catégorie (*Category*), famille (*Brand*) et type (*Type*). Ces paramètres étant organisés selon deux hiérarchies (*HBrand* et *HCateg*) ;
- La dimension date (D^{Date}) avec les paramètres date (*Date*), mois (*Month*) — accompagné du nom du mois en attribut faible (*Month_Name*) — et année (*Year*) ;
- La dimension fournisseur ($D^{Supplier}$) avec les paramètres code du fournisseur (*Supplier*) — accompagné du nom du fournisseur en attribut faible (*Name*) — ville (*City*), région (*Region*) et pays (*Nation*).

De ce schéma multidimensionnel noté E^{SSB}, des cuboïdes peuvent être spécifiés. Dans les exemples qui suivent, nous avons choisi de prendre l'intégralité des mesures du fait.

- $(F^{LineOrder}, \{D^{Customer}, D^{Date}, D^{Supplier}\})$ est un cuboïde correspondant au nœud CSD de la Figure 6,
- $(F^{LineOrder}, \{D^{Customer}, D^{Date}\})$ est un cuboïde correspondant au nœud CD de la Figure 6.

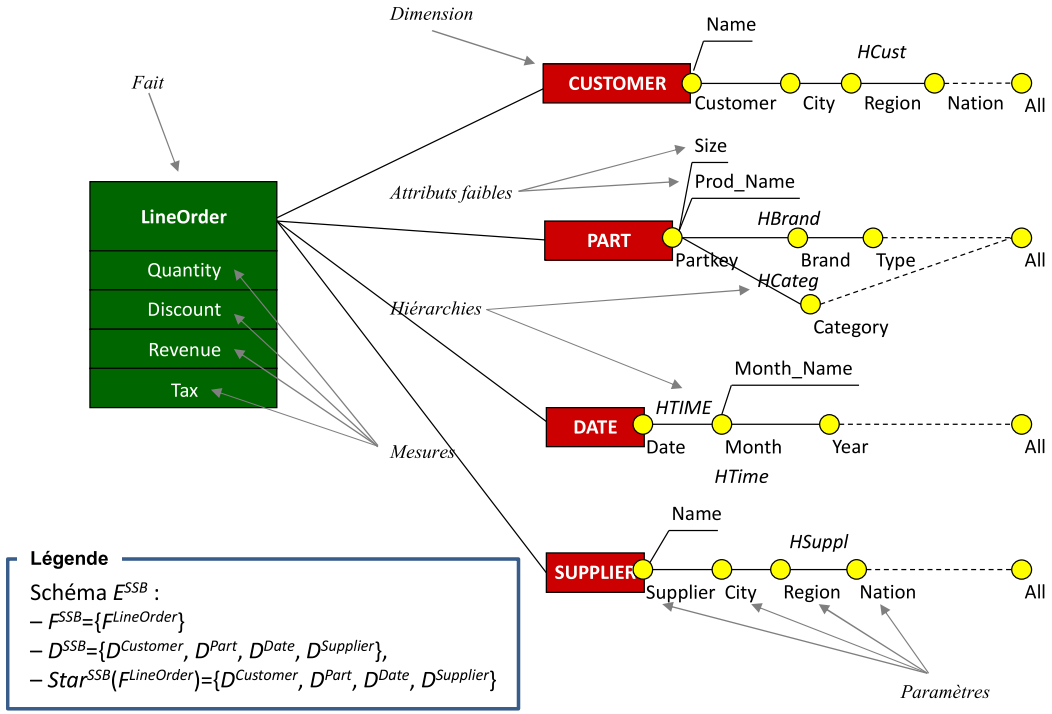


Figure 2 : Notations graphiques du modèle conceptuel multidimensionnel E^{SSB}.

4 Modélisation orientée documents d'un entrepôt de données multidimensionnelles

4.1 Formalisme pour du modèle orienté documents

Dans le modèle orienté documents, les données sont stockées en **collections** des **documents**. La structure des documents est définie par l'intermédiaire d'attributs (pouvant s'apparenter à des couples clé/valeur où l'attribut est la clé). Nous distinguons les **attributs simples** dont les valeurs sont atomiques, des **attributs**

composés dont les valeurs sont elles-mêmes des documents, appelées **documents imbriqués**. La structure d'un document peut se représenter comme un ensemble de chemins extraits de la structure arborescente formée par les attributs.

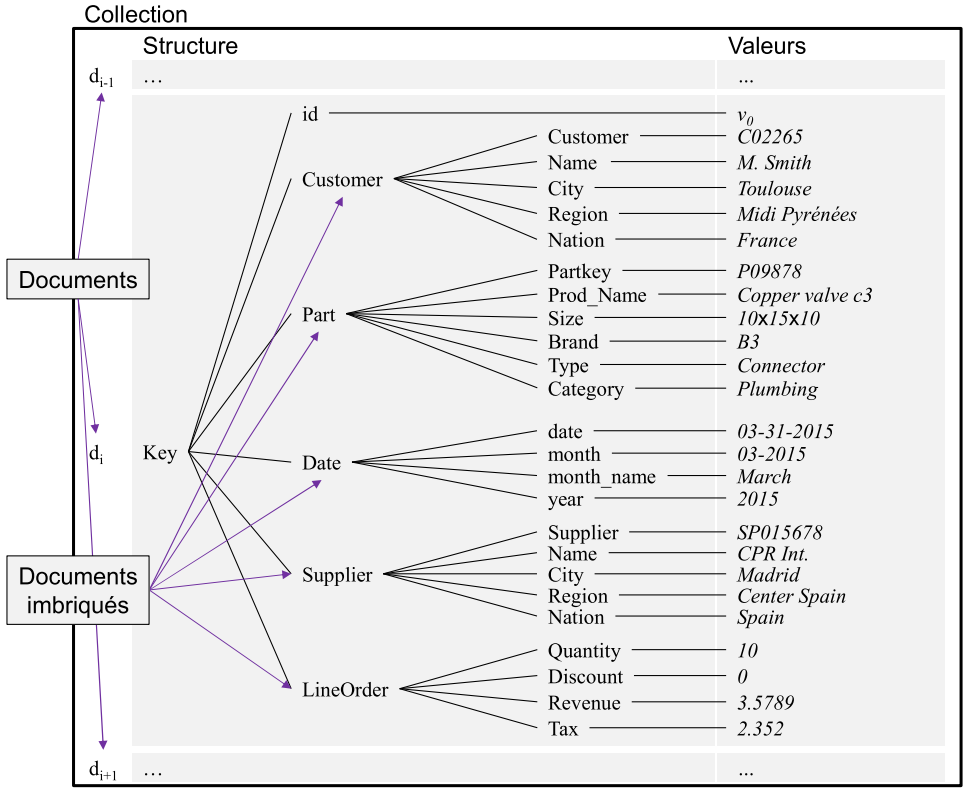


Figure 3 : Représentation arborescente des documents.

Un document est défini par un identifiant *key* et :

- C : la collection dans laquelle est le document,
- K : l'ensemble des identifiants de tous les attributs d'un document (identifiant du document ainsi que tous les noms des attributs, chacun servant d'identifiant pour la valeur qu'il contient),
- A : l'ensemble des attributs simples d'un document,
- V : l'ensemble des valeurs atomiques de tous les attributs d'un document (il s'agit de l'ensemble des valeurs des attributs de A),

- P : l'ensemble des chemins dans l'arbre du document. Un chemin $p \in P$ est décrit par $p=C.key.k_1.k_2 \ldots k_n.a$ où $k_i \in K$ sont les identifiants au sein du même chemin se terminant au nœud feuille $a \in A$ (les attributs simples sont forcément des nœuds feuilles) et *key* est l'identifiant du document.

L'exemple de la Figure 3 illustre le formalisme. Considérons l'arbre du document présenté en Figure 3 décrivant un document D^C d'une collection C avec un identifiant appelé *Key*. Un avantage important de cette représentation est la séparation strictre entre la représentation des éléments de structuration, et les valeurs.

4.2 Modèle orienté documents pour les entrepôts de données

Au sein des systèmes de stockage orientés documents, le modèle de données est déterminé non seulement par ses attributs et ses valeurs, mais aussi par le chemin pour atteindre les données, car les documents peuvent être imbriqués selon une organisation hiérarchique (cas des attributs composés). Au sein des systèmes orientés documents, l'exploitation de ce mécanisme d'imbrication offre la possibilité de plusieurs types d'implantations d'un entrepôt de données.

Dans cette section, nous présentons trois approches de modélisation orientée documents. Chaque approche repose sur un processus différent et elles correspondent à un ensemble (non exhaustif) d'approches possibles pour modéliser les sous parties d'un schéma multidimensionnel (une étoile, à savoir un fait et les dimensions qui lui sont associée). Dans la première approche MLD0, les étoiles correspondent à une unique collection à plat (les documents ne contiennent pas de documents imbriqués). Dans la seconde MLD1, nous utilisons toujours une unique collection par étoile, mais l'imbrication permet une meilleure expressivité en séparant les faits des dimensions. La troisième MLD2 distribue l'étoile dans plusieurs collections.

- **MLD0** : appelé modèle « **plat** », pour chaque fait, tous les attributs des dimensions associées et toutes les mesures du fait sont convertis en attributs dans un unique document sans sous-documents.
- **MLD1** : appelé modèle « **imbriqué** » Pour chaque fait, les attributs de chaque dimension sont des attributs contenus dans un sous-document représentant la dimension. Chaque mesure du fait suit la même règle et sont imbriqués dans un sous-document représentant le fait. Les documents du fait et des dimensions sont imbriqués dans un document représentant l'étoile. Ce modèle est inspiré de [2]. Notez qu'il existe de nombreuses manières d'imbriquer les documents modéliser les structures multidimensionnelles et ce modèle est l'une d'elles.
- **MLD2** : appelé modèle « **éclaté** » Pour chaque fait et ses dimensions associées, chaque structure est dans une collection dédiée : une par dimension et une pour le fait. Chaque collection est gardée simple : les documents ne contiennent pas de sous-documents. Les documents du fait contiennent également les attributs nécessaires pour pouvoir faire référence avec les documents des dimensions correspondantes. Ce modèle est connu pour avoir comme avantage un minimisation de l'espace disque et une meilleure intégrité des données, néanmoins, il peut ralentir l'interrogation des données.

4.3 Correspondances avec le modèle conceptuel

Le formalisme que nous avons défini précédemment pour décrire le modèle orienté documents nous permet de définir une correspondance entre notre modèle conceptuel multidimensionnel et chacun des modèles logiques précédents. Soit un cuboïde $O=(F^O,D^O)$ représentant un fait F^O (M^O étant ses mesures) associé à un ensemble de dimensions D^O.

La table suivante décrit les conversions des mesures m de F^O et des dimensions D de D^O dans chacun des trois modèles, *MLD0*, *MLD1* et *MLD2*. On pose

- C une collection générique,
- C^D une collection pour une dimension D,

- C^F une collection pour un fait F.

Table 1 Règles de passage du niveau conceptuel vers le niveau logique

Conceptuel	**Logique**		
	MLD0	**MLD1**	**MLD2**
$\forall D \in D^O, \forall a \in A^D$ (a est un attribut de D)	$a \rightarrow C.key.a$	$a \rightarrow C.key.N^D.a$	$a \rightarrow C^D.key.a \wedge$ $a=id^D \Rightarrow a \rightarrow C^F.key.a$ (*)
$\forall m \in M^O$	$m \rightarrow C.key.m$	$m \rightarrow C.key.N^F.m$	$m \rightarrow C^F.key.m$

(*) $D^{C^D}.key$ est différent de $D^{C^F}.key$ car il s'agit de documents de collections différentes (CD et CF)

Les transformations sont détaillées ci-après et présentent la structure des documents contenus dans les collections en fonction du modèle logique retenu.

Modèle conceptuel vers MLD0 : Pour implanter ce modèle depuis le modèle conceptuel multidimensionnel, nous appliquons les trois règles suivantes :

[1] Pour chaque cuboïde O (un fait F^O et ses dimensions associées D^O) une collection C est créée.
[2] Pour chaque mesure $m \in M^O$ un attribut simple m est créé, $C.key.m$.
[3] Pour chaque attribut (paramètre et attributs faibles) a de chaque dimensions $D \in D^O$, $a \in A^D$ un attribut simple a est créé, $C.key.a$.

Modèle conceptuel vers MLD1 : Pour implanter ce modèle depuis le modèle conceptuel multidimensionnel, nous appliquons les cinq règles suivantes :

[1] Pour chaque cuboïde O (un fait F^O et ses dimensions associées D^O) une collection C est créée.
[2] Pour chaque dimension D de D^O, un document imbriqué $C.key.N^D$ est créé.
[3] Pour chaque attribut $a \in A^D$ de chaque dimension D, un attribut simple a est créé dans le document imbriqué $C.key.N^D$ correspondant, tel que $C.key.N^D.a$.
[4] Pour F^O, un unique document imbriqué $C.key.N^F$ est créé.
[5] Pour chaque mesure $m \in M^O$ un attribut simple m est créé dans le document imbriqué $C.key.N^F$ tel que $C.key.N^F.m$.

Modèle conceptuel vers MLD2 : Pour implanter ce modèle depuis le modèle conceptuel multidimensionnel, nous appliquons les quatre règles suivantes :

[1] Pour chaque cuboïde O (un fait F^O et ses dimensions associées D^O), une collection C^F est crée.
[2] Pour chaque dimension D de D^O une collection C^D est créée.
[3] Pour chaque mesure $m \in M^O$ un attribut simple m est créé, $C^F.key.m$.
[4] Pour chaque attribut $a \in A^D$ de la dimension D, un attribut simple a est créé tel que $C^D.key.a$. De plus, si $a=id^D$ alors un attribut simple a est également créé dans C^F tel que $C^F.key.a$ (il s'agit des références vers les dimensions associées).

5 Expérimentations

Le but des expérimentations qui suivent est de valider l'instanciation d'un entrepôt de données multidimensionnelles avec les trois approches définies précédemment. Nous prenons également en compte la conversion d'un modèle logique à un autre, car ces conversions peuvent être nécessaires en fonction des

traitements à privilégier. Enfin, nous générons le cube OLAP et comparons le coût de chaque modèle. Pour les expérimentations, nous nous appuyons sur le Star Schema Benchmark, SSB [3],[14], dont un extrait de la structure des dimensions est présenté en Figure 2. Pour le stockage nous utilisons MongoDB, un des systèmes NoSQL orienté documents les plus répandus actuellement.

5.1 Protocole expérimental

Données. Les données sont générées via SSB+ [3] (permettant d'utiliser les avantages du NoSQL tel que le paradigme Map/Reduce). Ce benchmark, dérivé de SSB [14], contient un fait *LineOrder* et quatre dimensions (*Customer*, *Part*, *Date* et *Supplier*). Chaque dimension est composée de nombreux attributs qui sont organisés dans une ou plusieurs hiérarchies. Avec SSB+ nous générons automatiquement des données brutes dans un format JSon, adapté au chargement dans MongoDB. Nous employons différents facteurs d'échelle de génération des données (*scale factor*, *sf*), plus précisément, *sf*=1, *sf*=10, *sf*=25 (que nous noterons SF1, SF10 et SF25 par la suite). Le facteur d'échelle SF1 génère 10^7 lignes dans le fait LineOrder ; dix fois plus pour SF10 et ainsi de suite.

Chargement des données. Les données sont chargées dans MongoDB en utilisant ses instructions propres. Ces dernières permettent de charger les données plus rapidement à partir de fichiers. La version actuelle de MongoDB que nous utilisons charge les données à partir du format JSon.

Calcul du cube OLAP (ou treillis d'agrégats). Pour calculer le treillis d'agrégats, nous utilisons le « *aggregation pipeline* » recommandée comme étant la meilleure alternative en termes de performance par MongoDB. Quatre niveaux d'agrégats sont calculés à partir du niveau de base fourni par le générateur du benchmark. Ces agrégats sont :

- toutes les combinaisons de trois dimensions,
- toutes les combinaisons de deux dimensions,
- toutes celles de une dimension et enfin
- toutes les données.

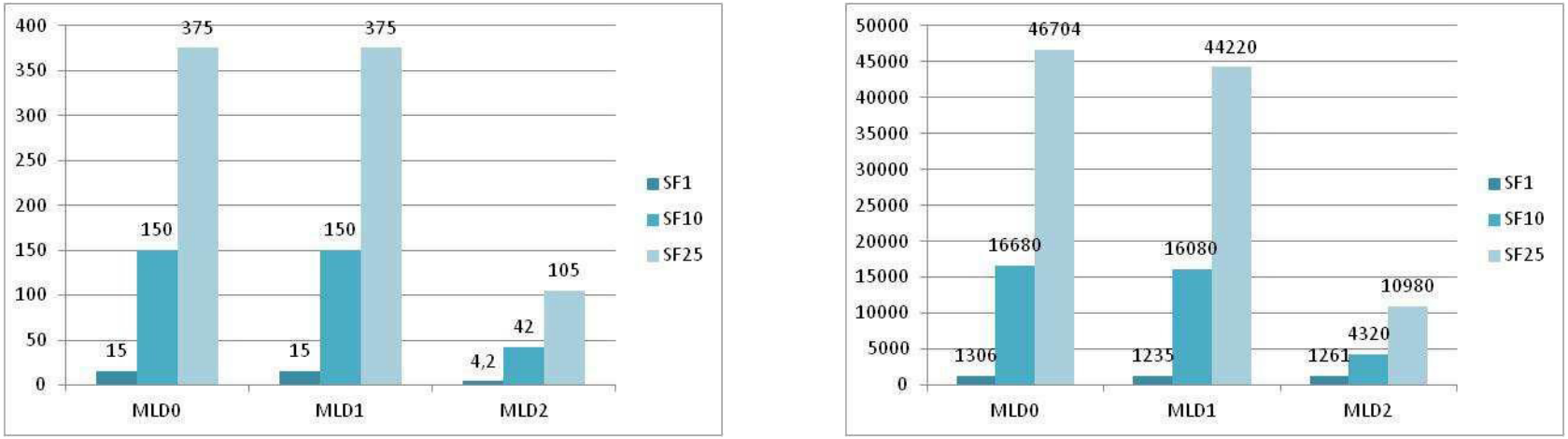
A chaque niveau les fonctions d'agrégation *Max*, *Min*, *Sum* et *Count* sont appliquées sur chaque mesure.

Infrastructure. Les expérimentations sont faites sur un cluster de 3 PC standards (CPU i5 quadri cœurs, 8Go de RAM, 2x2To de disque et 1Gb/s de réseau). Chaque PC étant un nœud de données et l'un d'eux agit en outre comme maître, gérant la répartition des données et des traitements (*sharding*).

5.2 Résultats

La Figure 4 présente les résultats des temps de chargement des données en fonction du modèle logique utilisé et du facteur d'échelle. Nous observons qu'avec SF1 nous avons 10^7 lignes dans le fait pour un total de 4,2 Go d'espace disque pour MLD2 (ce modèle évitant au maximum la redondance il est normal qu'il présente un espace occupé minimal) ; par opposition, 15 Go sont nécessaires pour MLD0 et MLD1. Pour SF10 et SF25 nous avons $10x10^7$ et $25x10^7$ lignes dans le fait, soit 42 Go pour MLD2 (150 Go pour MLD0 et MLD1) et 105 Go pour MLD2 (375 Go pour MLD0 et MLD1) en occupation d'espace disque.

Dans le modèle MLD2, les collections représentant respectivement les dimensions *Customers*, *Suppliers*, *Part* et *Date* sont constituées respectivement de 50 000, 3 333, 3 333 333 et 2 556 enregistrements.



(a) *Volume en Go* (b) *Temps de chargement en secondes*

Figure 4 : Volume et temps de chargement des données dans MongoDB en fonction du modèle logique.

Dans la Figure 5 nous présentons le temps nécessaire pour passer d'un modèle logique à l'autre avec le facteur d'échelle SF1 (opérations sur 10^7 enregistrements). Lors de la conversion du modèle MLD0 vers MLD1 et vice-versa, les temps sont comparables. Ceci s'explique par le fait que pour transformer les données du format MLD0 vers MLD1, un ajout d'un niveau dans la collection est effectué (un document imbriqué est inséré en tant que niveau intermédiaire dans la structure hiérarchisée des documents), l'opération inverse consiste à enlever ce niveau intermédiaire. La conversion est plus compliquée lorsque l'on considère MLD0 et MLD2 : il est nécessaire d'éclater les données en plusieurs collections : pas moins de 5 projections sont appliquées sur les données du MLD0 ainsi qu'une sélection distinctes de chaque identifiant de chaque ligne dans chaque dimension. Bien que le volume les données produites soit moindre, ce traitement requiert beaucoup plus de temps de calcul pour générer le format MLD2. La conversion inverse, MLD2 vers MLD0, est de loin le processus le plus lent (temps de génération entre 5h et 125h pour SF1). Ceci est dû au fait que la plus part des systèmes NoSQL (MongoDB inclus) supporte mal (nativement) les jointures. Il nous a été nécessaire de tester différentes techniques d'optimisation programmées manuellement.

Figure 5 : Temps de conversion dans MongoDB d'un modèle logique vers un autre avec SF1.

En Figure 6, nous résumons nos observations concernant le calcul du cube OLAP en utilisant les données du modèle MLD0 avec un facteur SF1 (10^7 enregistrements). Chaque nœud du treillis représente un cuboïde et correspond à une combinaison de dimensions. Nous indiquons le temps nécessaire pour calculer

le cuboïde ainsi que le nombre d'enregistrements qu'il contient. Nous calculons les agrégations en utilisant un paramètre de chaque dimension intermédiaire dans la hiérarchie générant ainsi une agrégation avec moins d'enregistrements que si nous avions pris un niveau de détails plus bas dans la hiérarchie.

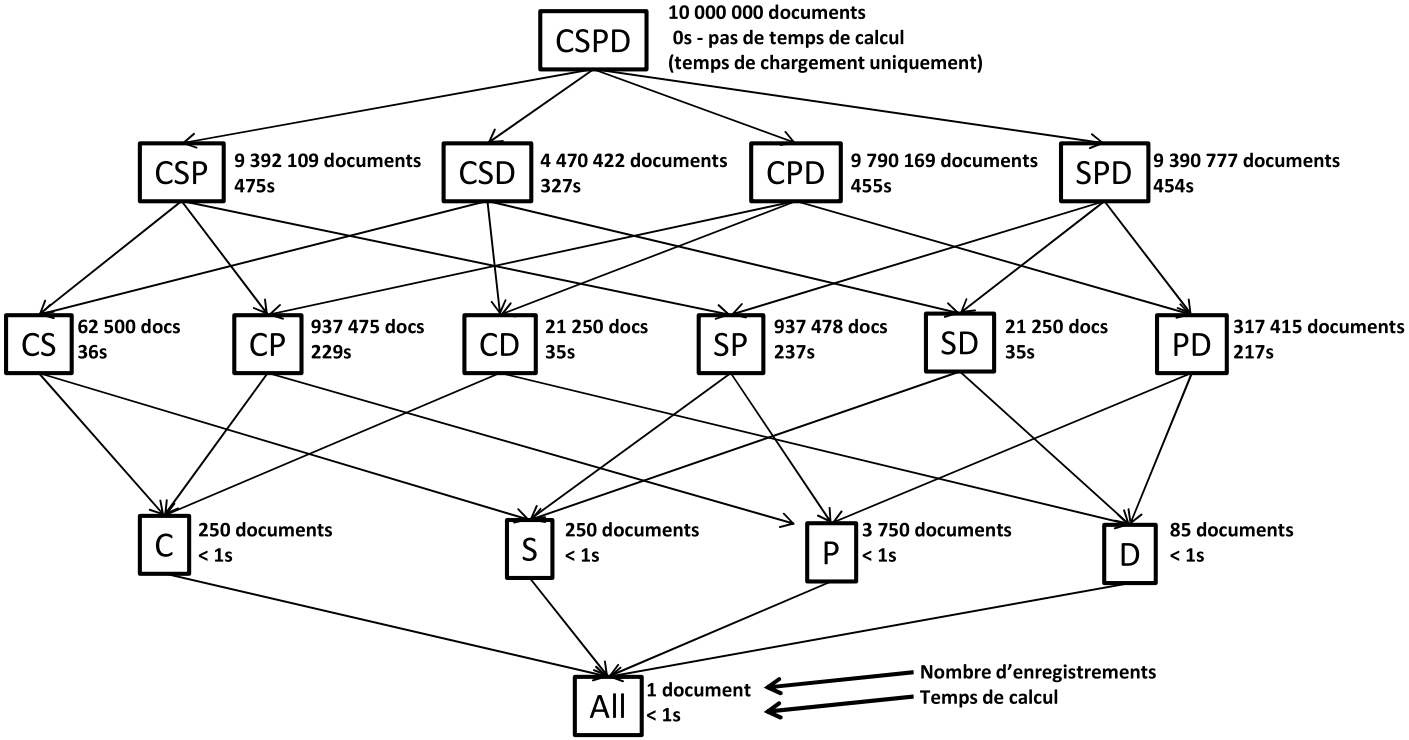


Figure 6 : Temps de calcul et nombre d'enregistrement de chaque cuboïde OLAP.

Comme attendu, nous observons une diminution du nombre d'enregistrements au fur et à mesure que nous augmentons la « taille » des agrégats (c'est-à-dire que nous agrégeons de plus en plus de valeurs en une seule). La même observation est également faite pour les temps de calcul. Nous avons besoin entre 300 et 500 secondes pour calculer les cuboïdes du premier niveau (combinaisons de 3 dimensions) ; entre 30 et 250 secondes pour le second niveau (combinaisons de 2 dimensions) ; et moins de 1 seconde pour les troisième et quatrième niveaux (combinaisons de 1 et 0 dimensions).

Le même processus avec le modèle MLD1 fournit des résultats similaires (résultats non présentés en détails faute de place), mais les performances se dégradent avec le modèle MLD2 en raison des jointures nécessaires. L'impact est cependant limité au premier niveau du treillis car les trois autres niveaux peuvent systématiquement être calculés à partir du niveau précédent déjà calculé. Il est important de noter que ceci est possible car les quatre fonctions d'agrégation que nous employons sont distributives [11].

Observations : nous observons que nous avons des temps comparables pour charger les données dans un modèle puis pour les convertir dans un autre (exception faite de la conversion MLD2 vers MLD0). Nous observons également des temps « raisonnables[3] » pour le calcul des cuboïdes OLAP. Ces observations sont importantes car, d'une part, elles nous prouvent que nous pouvons instancier des entrepôts de données multidimensionnelles dans un environnement NoSQL orienté documents ; et, d'autre part, nous pouvons envisager l'utilisation de modèles pivots ainsi que d'agrégats calculés en parallèle via les processus typiques du NoSQL et implanté en utilisant un modèle logique plus spécifique.

6 Conclusion

Dans cet article nous avons étudié l'instanciation d'entrepôts de données multidimensionnelles avec un système NoSQL orienté documents. Nous avons proposé trois approches logiques. En utilisant un formalisme simple, nous décrivons le processus de transformation depuis notre modèle conceptuel multidimensionnel vers chaque modèle logique. En outre, nous permettons une conversion des données entre les différentes modélisations logiques.

Nos expérimentations montrent l'instanciation d'un entrepôt de données avec chacune des trois approches. Le modèle éclaté (MLD2) utilise moins d'espace disque mais est couteux en temps d'exécution de requêtes en raison des jointures nécessaires. Les modèles simples MLD0 (modèle plat) et MLD1 (modèle imbriqué) ne montrent pas d'écarts significatifs en termes de performances entre eux. Passer d'un modèle à l'autre est comparable en temps de chargement des données. Une conversion est toutefois peu performante ; elle correspond à la fusion de plusieurs collections de documents (MLD2) vers une unique collection. Des résultats intéressants ont également été obtenus lors du calcul du treillis d'agrégats (ou cube OLAP) et de son stockage dans l'environnement orienté documents.

Plusieurs perspectives sont envisagées. Nous allons considérer d'autres modèles logiques NoSQL, le modèle orienté colonnes et le modèle orienté graphes et les comparer au modèle orienté documents. Après avoir fait ces explorations, il nous sera nécessaire de disposer d'un formalisme de modèle logique assez expressif pour être compatible avec ces différentes plateformes afin de spécifier de manière générique différentes implantations et les transformations nécessaires pour passer de l'une à l'autre.

7 Remerciements

Ces travaux ont été réalisés avec le soutien de l'ANRT (Association Nationale de la Recherche et de la Technologie) dans le cadre d'une collaboration CIFRE avec la société Capgemini.

8 Bibliographie

[1] CHAUDHURI, S., DAYAL, U., *An overview of data warehousing and OLAP technology*, SIGMOD Record, 26(1), ACM, p. 65-74, 1997

[2] CHEVALIER, M., EL MALKI, M., KOPLIKU, A., TESTE, O., TOURNIER, R., *Implementing Multidimensional Data Warehouses into NoSQL*, 17th International Conference on Entreprise Information Systems (ICEIS), Barcelona, Spain, 2015.

[3] Nous jugeons les temps de calcul raisonnables pour un environnement de type Mégadonnées / Big Data exécuté sur du matériel standard et sans le réseau à base de fibres optiques recommande (à savoir un réseau à 10 000 Gb/s).

[3] CHEVALIER, M., EL MALKI, M., KUPLIKU, A., TESTE, O., TOURNIER, R., *Benchmark for OLAP on NoSQL Technologies, Comparing NoSQL Multidimensional Data Warehousing Solutions*, 9th Int. Conf. on Research Challenges in Information Science (RCIS), IEEE, 2015.

[4] CHEVALIER, M., EL MALKI, M., KUPLIKU, A., TESTE, O., TOURNIER, R., *Entrepôts de données multidimensionnelles NoSQL*. 11ème Journées francophones sur les Entrepôts de Données et l'Analyse en ligne (EDA'15), p.161-176, Bruxelles, Belgique, 2015.

[5] COLLIAT, G., *OLAP, Relational, and multidimensional database systems*, SIGMOD Record, 25(3), ACM, p. 64.69, 1996

[6] CUZZOCREA, A., SONG, I.Y., DAVIS, K.C., *Analytics over large-scale multidimensional data: The big data revolution!* 14th International Workshop on Data Warehousing and OLAP (DOLAP), ACM, p. 101-104, 2011

[7] DEDE, E., GOVINDARAJU, M., GUNTER, D., CANON, R.S., RAMAKRISHNAN, L., *Performance evaluation of a MongoDB and Hadoop platform for scientific data analysis*, 4th ACM Workshop on Scientific Cloud Computing (Science Cloud), ACM, p. 13-20, 2013

[8] DEHDOUH, K., BOUSSAID, O., BENTAYEB, F., *Columnar nosql star schema benchmark*. Model and Data Engineering, LNCS 8748, Springer, p. 281-288, 2014

[9] GOLFARELLI, M., MAIO, D., RIZZI, S., *The dimensional fact model: A conceptual model for data warehouses*. International Journal of Cooperative Information Systems (ijCIS), 7(2-3), World Scientific Publishing p. 215-247, 1998

[10] 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, Cross-Tab, and Sub Totals*, Data Mining and Knowledge Discovery, 1(1), Kluwer Academic, p. 29-53, 1997.

[11] HASSAN, A., RAVAT, F., TESTE, O., TOURNIER, T., ZURFLUH, G., *Differentiated Multiple Aggregations in Multidimensional Databases*. 14th International Conference on Data Warehousing and Knowledge Discovery (DAWAK'12), p.93-104, Vienna (Austria), September 2012

[12] KIMBALL, R., ROSS, M., *The Data Warehouse Toolkit: The Definitive Guide to Dimensional Modeling,* 3e ed., John Wiley & Sons, 2013

[13] MIOR, M.J., *Automated Schema Design for NoSQL Databases*, SIGMOD PhD symposium, ACM, p. 41-45, 2014

[14] ONEIL, P., ONEIL, E., CHEN, X., REVILAK, S., *The star schema benchmark and augmented fact table indexing*, Performance Evaluation and Benchmarking, LNCS 5895, Springer, p. 237-252, 2009

[15] RAVAT, F., TESTE, O., TOURNIER, R., ZURUH, G., *Algebraic and graphic languages for OLAP manipulations*, International Journal of Data Warehousing and Mining (ijDWM), 4(1), Idea Group Publishing, p. 17-46, 2008

[16] RAVAT, F., TESTE, O., TOURNIER, R., ZURUH, G., *Graphical Querying of Multidimensional Databases.* East-European Conference on Advances in Databases and Information Systems (ADBIS'07), LNCS 4690, p.298-313, Varna (Bulgaria), September 2007

[17] STONEBRAKER, M., *New opportunities for New SQL*, Communications of the ACM, 55(11), ACM, p. 10-11, 2012

[18] ZHAO, H., YE, X., *A practice of TPC-DS multidimensional implementation on NoSQL database systems*, Performance Characterization and Benchmarking, LNCS 8391, Springer, p. 93-108, 2014