Modélisation des métadonnées d'un *data lake* en *data vault*

Iuri D. Nogueira*, Maram Romdhane*, Jérôme Darmont*

* Université de Lyon, Lyon 2, ERIC EA 3083
5 avenue Pierre Mendès France, F69676 Bron Cedex
iuri.deolindonogueira@univ-lyon2.fr, maram.romdhane@univ-lyon2.fr,
jerome.darmont@univ-lyon2.fr

Résumé. Avec l'avènement des mégadonnées (*big data*), l'informatique décisionnelle a dû trouver des solutions pour gérer des volumes et une variété de données plus grands encore que dans les entrepôts de données, qui se sont révélés mal adaptés. Les lacs de données (*data lakes*) répondent à ces besoins du point du vue du stockage, mais nécessitent la gestion de métadonnées adéquates pour garantir un accès efficace aux données. Sur la base d'un modèle multidimensionnel de métadonnées conçu pour un lac de données patrimoniales présentant un défaut d'évolutivité de schéma, nous proposons dans cet article l'utilisation de la modélisation ensembliste, et plus particulièrement d'un *data vault*, pour traiter ce problème. Pour montrer la faisabilité de cette approche, nous instancions notre modèle conceptuel de métadonnées en modèles logiques et physiques relationnel et orienté document, respectivement. Nous comparons également les modèles physiques en termes de stockage et de temps de réponse aux requêtes sur les métadonnées.

Cet article est une version longue de celui qui a été publié en version courte dans les actes d'EGC 2018.

1 Introduction

Les lacs de données (*data lakes*) ont été introduits par Dixon (2010). Ils proposent une manière, née avec les mégadonnées (*big data*), de stocker dans leur format natif des données volumineuses, variées et diversement structurées, dans un lieu de stockage évolutif en vue de les analyser (*reporting*, visualisation, fouille de données...) (Clapeau, 2017). C'est un concept qui s'oppose au processus d'entreposage de données, qui aboutit à une base de données décisionnelle intégrée, très structurée et orientée sur un sujet précis, mais qui a l'inconvénient de diviser les données en silos étanches (Stein et Morrison, 2014).

Toutefois, tout le monde s'accorde pour dire qu'un lac de données doit être bien conçu sous peine de devenir un marécage (*data swamp*) inexploitable (Alrehamy et Walker, 2015) ; c'est à dire qu'il doit permettre le requêtage des données (sélection/restriction) avec un bon temps de réponse et pas seulement leur stockage et leur accès « clé-valeur ». En revanche, les solutions pour y parvenir sont peu ou prou inexistantes dans la littérature et relèvent à l'heure actuelle de pratiques industrielles peu divulguées.

C'est pourquoi Pathirana (2015) a proposé un modèle conceptuel de métadonnées permettant l'indexation et l'interrogation efficace d'un lac de données patrimoniales (données XML, textes, plans et photos...). Ce modèle multidimensionnel est proche des modèles en flocons de neige en usage dans les entrepôts de données, mais ne concerne que les métadonnées et non le corpus de documents lui-même.

Ce modèle de métadonnées a été instancié au niveau physique dans différents systèmes de gestion de bases de données (SGBD) NoSQL pour permettre le passage à l'échelle. Cependant, le type de schéma employé est très difficile à faire évoluer lorsque ceux des sources de données évoluent ou que de nouvelles sources sont à prendre en compte, alors que c'est un point crucial dans la gestion d'un lac de données.

En conséquence, nous proposons dans cet article de :

1. remplacer le modèle multidimensionnel de Pathirana (2015) par un modèle ensembliste, en l'occurrence un *data vault* (Linstedt, 2011; Hultgren, 2012; Linstedt et Olschimke, 2015), qui est un modèle de données permettant des évolutions de schémas aisées et qui n'a, à notre connaissance, jamais été employé dans le contexte de la gestion de métadonnées ;
2. vérifier la faisabilité, d'une part, et l'efficacité de ce modèle en termes de réponse aux requêtes sur les métadonnées (à la manière d'un index), d'autre part, car il induit de nombreuses jointures. Pour cela, nous traduisons notre *metadata vault* conceptuel en différents modèles logiques (relationnel et orienté document) et physiques (PostgreSQL et MongoDB), ce qui permet également de comparer l'efficacité respective des deux modèles physiques.

Le reste de cet article est organisé comme suit. La Section 2 est consacrée à un état de l'art concernant, d'une part, les lacs de données et la gestion des métadonnées et, d'autre part, la modélisation ensembliste et plus particulièrement les *data vaults*. La Section 3 présente notre modèle conceptuel de métadonnées en *data vault*, ainsi que ses déclinaisons logiques et physiques. La Section 4 détaille les expériences que nous avons menées pour valider la faisabilité de notre approche et comparer les modèles physiques PostgreSQL et MongoDB en termes de stockage et de temps de réponse aux requêtes sur les métadonnées. Finalement, la Section 5 conclut cet article et annonce quelques perspectives de recherche.

2 État de l'art

2.1 Lacs de données et métadonnées

Les lacs de données sont généralement construits pour intégrer de très grands volumes de données non structurées de manière rapide. Ils ne se limitent toutefois pas à une technologie de stockage (la plupart du temps HDFS – *Hadoop Distributed File System*), mais proposent un nouvel écosystème de données permettant rapidement, à la demande, de croiser des données (Ganore, 2015), sans besoin de prétraitements coûteux tels que la construction d'un entrepôt de données. Les données sont immédiatement accessibles, contrairement, de nouveau, aux entrepôts de données qui sont rafraîchis périodiquement via une phase d'ETL (extraction, transformation, chargement) qui peut être coûteuse (Miloslavskaya et Tolstoy, 2016).

Les lacs de données ont une architecture « plate », où chaque élément de données possède un identifiant unique et un ensemble d'étiquettes (*tags*) le caractérisant (Miloslavskaya et Tolstoy, 2016). Par conséquent, ils ne nécessitent pas de schéma rigide comme les entrepôts de données. De plus, les étiquettes caractérisant les données sont des métadonnées indispensables à la compréhension des donnée et à leur accès (Alrehamy et Walker, 2015), sans même parler de l'efficacité des requêtes.

L'avantage de l'étiquetage des données est que de nouvelles données, de nouvelles sources, peuvent être insérées au fil de l'eau. Une fois les données étiquetées, elles doivent simplement être connectées aux données déjà stockées. Ce dispositif permet aux utilisateurs de formuler des requêtes directement, sans besoin de recourir à l'aide d'un expert en informatique décisionnelle.

2.2 Modélisation ensembliste et *data vaults*

La modélisation ensembliste (*ensemble modeling*) est une approche qui date du début des années 2010 et qui vise à renormaliser les entrepôts de données afin de les rapprocher des concepts métiers qu'ils modélisent et de permettre une meilleure évolutivité, tant en termes de données que de schéma (Rönnbäck et Hultgren, 2013).

Les deux approches qui se dégagent dans cette mouvance sont les *data vaults* et l'*anchor modeling* (Regardt et al., 2009; Rönnbäck et al., 2010). Elles sont en fait très proches (Rönnbäck et Hultgren, 2013), avec une évolutivité un peu plus aisée et une évolution de schéma non destructive pour l'*anchor modeling*, mais un plus grand nombre d'objets à gérer en raison d'une modélisation en sixième forme normale (6NF), ainsi que des procédures de maintenance des attributs temporels (*timestamps*) non automatisées. Les *data vaults* sont plus proches de la modélisation multidimensionnelle traditionnelle (ils adoptent la troisième forme normale – 3NF) et sont supportés par un plus grand nombre d'outils, ce qui nous a fait privilégier ces derniers dans ce travail. Ce choix n'est toutefois pas définitivement arrêté.

Les *data vaults* ont été inventés par Linstedt (2011) pour répondre aux besoins de l'industrie. Un *data vault* est défini au niveau logique relationnel comme un ensemble lié de tables normalisées orienté détail et suivi d'historique, qui prend en charge un ou plusieurs domaines fonctionnels d'une organisation. C'est une approche hybride englobant la 3NF et le schéma en étoile (Linstedt, 2011; Hultgren, 2012; Linstedt et Olschimke, 2015).

Cette architecture permet la construction incrémentale du schéma à moindre coût (ce qui rend facile la modification de règles métier), l'intégration transparente de données non structurées, le chargement de données en quasi-temps réel et une capacité de passage à l'échelle permettant la gestion de mégadonnées de l'ordre de plusieurs pétaoctets (Linstedt et Olschimke, 2015). Enfin, les méthodes d'intelligence artificielle et de fouille de données peuvent aisément être appliquées directement sur un *data vault* (Linstedt, 2015).

Plus concrètement, un *data vault* est composé des éléments principaux suivants (Linstedt, 2011; Hultgren, 2012; Linstedt et Olschimke, 2015).

— Un centre (*hub*) est une entité de base qui représente un concept métier (un niveau hiérarchique de dimension dans un entrepôt de données classique, client ou produit, par exemple). Il contient principalement une clé (*business key*).
— Un lien (*link*) matérialise une association entre deux centres ou plus. Il correspondrait à une entité de faits dans un entrepôt classique.

— Un satellite contient les différentes données descriptives (attributs) relatives à un centre ou un lien.

Notons finalement que, bien que décrits au niveau logique (relationnel) par Linstedt (2011), ces concepts peuvent facilement être transposés au niveau conceptuel (Jovanovic et Bojicic, 2012). Un exemple de modèle conceptuel de *data vault* est présenté en Figure 1.

3 Modèle de métadonnées en *data vault*

3.1 Corpus de données

Afin de proposer un cas d'utilisation susceptible d'illustrer notre propos et de servir de preuve de concept, nous avons exploité le corpus de données issu du projet TECTONIQ, qui vise à valoriser le patrimoine industriel textile de Lille Métropole (Kergosien, 2017). Ce corpus rassemble des données hétérogènes, fournies par différentes sources (Table 1) et doit permettre divers types d'analyses. C'est pourquoi le stockage du corpus est effectué dans un lac de données et que Pathirana (2015) a proposé une modélisation multidimensionnelle de ses métadonnées. Nous nous proposons dans cet article de rendre ces métadonnées évolutives face à l'ajout de nouvelles sources de données, en détournant en quelque sorte les concepts des *data vaults* (centres, liens et satellites) pour stocker les métadonnées.

TAB. 1 – *Composition du corpus TECTONIQ*

Source	**Inventaire**	**La Voix du Nord**
Description	Descriptions de bâtiments industriels	Articles de presse liés à l'industrie textile
Nombre de documents	49	1
Nombre d'instances	49	30
Taille globale	120 Ko	92 Ko
Format	XML (données)	XML (documents)
Source	**Images IHRIS**	**Ouvrages**
Description	Photos et plans de bâtiments et monuments liés à l'industrie textile	Livres d'histoire de France du domaine public
Nombre de documents	30	165
Nombre d'instances	30	165
Taille globale	2,78 Mo	1011,25 Mo
Format	JPEG	PDF

3.2 Modèle conceptuel

La Figure 1 illustre notre modèle de métadonnées, qui exploite la modélisation conceptuelle des concepts logiques relationnels des *data vaults* proposée par Jovanovic et Bojicic (2012). Les rectangles arrondis bleus y représentent les centres (*hubs*), les rectangles gris les satellites et l'hexagone vert un lien (*link*). Les associations entre centres et liens sont toutes

de cardinalité « plusieurs à plusieurs »pour assurer la plus grande généralité, tandis que les associations entre centres ou liens et satellites sont de cardinalité « un à plusieurs ».

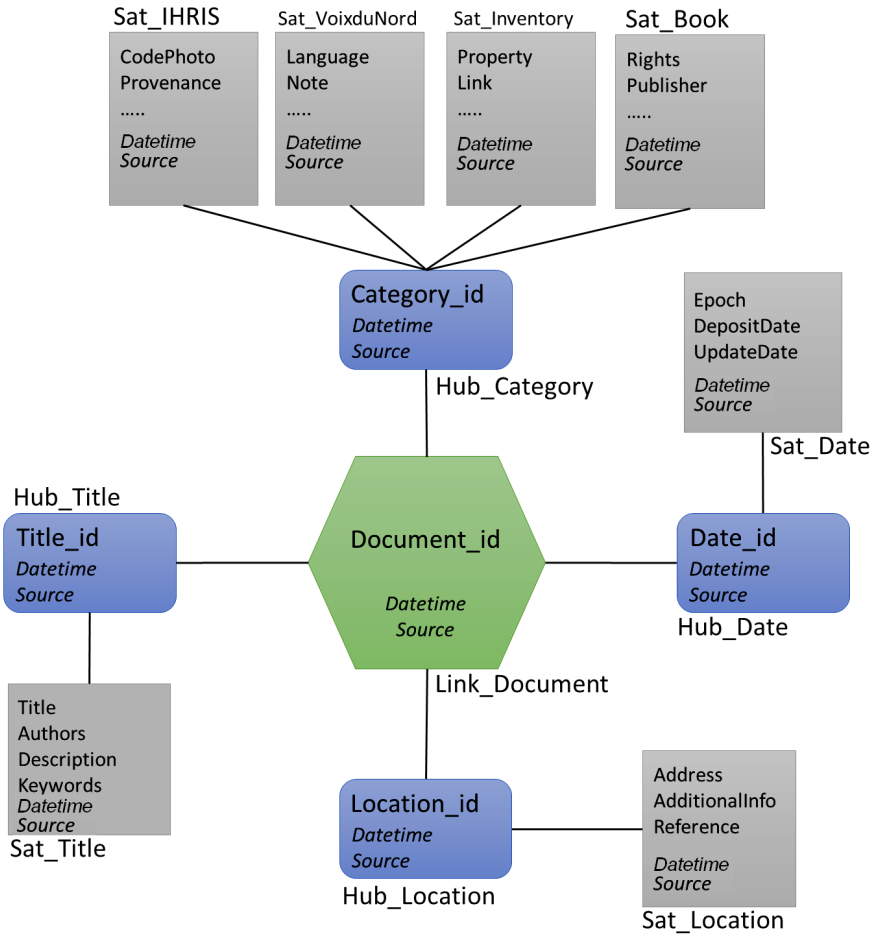

FIG. 1 – *Modèle de métadonnées en data vault*

Afin d'identifier les centres, qui sont les entités les plus importantes et les plus utilisées pour les requêtes, nous ciblons les métadonnées indicatrices des caractéristiques de chaque document en entrée, qui sont susceptibles de constituer des clés. Ainsi, nous sélectionnons le titre (*Hub_Title*), la localisation (*Hub_Location*), la date (*Hub_Date*) et la catégorie des documents (*Hub_Category*).

À chacun de ces centres, nous associons ensuite un satellite qui contient les attributs descriptifs du centre. Par exemple, le satellite connecté au centre *Hub_Title*, *Sat_Title*, contient les attributs titre (*Title*), auteurs (*Authors*), description (*Description*) et mots clés (*Keywords*).

Le centre *Hub_Category* est associé à quatre satellites qui forment une classification correspondant à chacune des source de données dont nous disposons. Les attributs descriptifs de

ces satellites sont spécifiques à chacune des sources.

Le lien document (*Link_Document*) permet d'associer tous les centres.

Finalement, puisque notre modèle concerne seulement des métadonnées, chaque entité (centre, lien, satellite) est décrite par une référence directe (*Source*) à un document (fichier physique) du lac de données. C'est par cette référence que passe tout accès aux données.

Grâce à cette modélisation, toute nouvelle source de données ou évolution de schéma au sein du corpus de données peut être prise en charge par l'ajout de satellites, pour les cas les plus simples, voire de centres et de liens. De plus, les entités qui deviendraient obsolètes sont simplement identifiées grâce à leur horodatage (attribut *Datetime*).

3.3 Modèles logiques et physiques

Afin de montrer que notre modèle conceptuel peut s'adapter à différents contextes, ainsi que de les comparer, nous le déclinons en deux types de modèles logiques et physiques : un modèle relationnel avec PostgreSQL et un modèle NoSQL orienté document avec MongoDB.

Le modèle relationnel des métadonnées (Table 2) [1] est traduit du modèle conceptuel (Figure 1) de manière classique. Notons seulement que, dans la Table 2, les éléments de la colonne attributs postfixés par $\_id$ sont des clés étrangères, et que les clés primaires des satellites sont également des clés étrangères qui référencent la clé primaire du centre correspondant. Enfin, les attributs $Datetime$ et $Source$ présents dans toutes les tables ne sont pas répétés par souci de clarté, sauf quand ils font partie d'une clé.

TAB. 2 – *Modèle logique relationnel*

Table	**Clé primaire**	**Attributs**
Centres		
Hub_Title	Title_id	
Hub_Date	Date_id	
Hub_Location	Location_id	
Hub_Category	Category_id	
Lien		
Link_Document	Document_id	Title_id, Location_id, Date_id, Category_id
Satellites		
Sat_Title	Title_id, Datetime	Title, Authors, Description, Keywords
Sat_Date	Date_id, Datetime	Epoch, DepositDate, UpdateDate
Sat_Location	Location_id, Datetime	Address, AdditionalInfo, Reference
Sat_IRHIS	Category_id, Datetime	CodePhoto, Provenance
Sat_VoixDuNord	Category_id, Datetime	Language, Note
Sat_Inventory	Category_id, Datetime	Property, Link
Sat_Book	Category_id, Datetime	Rights, Publisher

Le modèle orienté document des métadonnées (Table 3) exploite la notion de collection. Chaque centre, lien et satellite est traduit en collection comportant divers documents ayant cha-

1. Nous adoptons cette présentation inhabituelle en tableau pour uniformiser la représentation avec le modèle orienté document (Table 3) et faciliter la comparaison des deux modèles logiques.

cun un identifiant ($ObjectID$). Dans le modèle physique de MongoDB, cet identifiant est généré automatiquement à la création d'un document. Comme dans le cas relationnel, les champs $Datetime$ et $Source$ présents dans toutes les collections ne sont pas répétés par souci de clarté.

TAB. 3 – *Modèle logique orienté document*

Collection	**ObjectID**	**Champs**
Centres		
Hub_Title	Title_id	
Hub_Date	Date_id	
Hub_Location	Location_id	
Hub_Category	Category_id	
Lien		
Link_Document	Document_id	
Satellites		
Sat_Title	Title_id	Title, Authors, Description, Keywords
Sat_Date	Date_id	Epoch, DepositDate,UpdateDate
Sat_Location	Location_id	Address, AdditionalInfo, Reference
Sat_IRHIS	Category_id	CodePhoto, Provenance
Sat_VoixDuNord	Category_id	Language, Note
Sat_Inventory	Category_id	Property, Link
Sat_Book	Category_id	Rights, Publisher

Chaque centre est associé au lien $Link\_Document$ et à son ou ses satellites grâce à son $ObjectID$. Par exemple, Hub_Title est associé à $Link\_Document$ et à Sat_Title grâce à $Title\_id$. Enfin, la Figure 2 présente un exemple de traduction du modèle logique orienté document en modèle physique MongoDB au format JSON (*JavaScript Object Notation*).

Au niveau physique, les métadonnées sont extraites des documents sources par un processus ETL, via des scripts.

4 Validation expérimentale

L'objectif de cette section est de montrer la faisabilité de notre modèle de métadonnées en *data vault* et de comparer les modèles physiques de la Section 3.3 en termes d'espace de stockage et de temps de réponse aux requêtes.

Nous avons mené nos expériences sur un PC Intel Core i5-5300U 2,30 GHz avec 8 Go de mémoire, sous Windows 64 bits. Les versions des SGBD sont PostgreSQL 9.6.2-1 et MongoDB ssl 3.4.3. Le corpus de données que nous avons utilisé comme jeu de test est celui de la Table 1.

4.1 Volume de stockage

Après avoir inséré les métadonnées dans le format natif des SGBD choisis, nous avons mesuré le volume de stockage nécessaire dans les deux cas (Table 4). Nos mesures tiennent

```
{
 "Sat_Book": {
  "00000C842_001": {
   "dc:relation": "http://www.sudoc.fr/122661389",
   "dc:creator": [
    "Petit, Jules",
    "Chambre de commerce et d'industrie (Boulogne-sur-Mer, Pas-de-Calais).",
    "Commission du projet de chemin de fer direct de Calais à Marseille"
   ],
   "dc:subject": "Calais-Marseille, Chemin de fer (France).",
   "dc:publisher": "Villeneuve d'Ascq : SCD Lille 3",
   "dc:date": "[2008]",
   "dc:format": "application/pdf",
   "dc:language": "fre",
   "dc:rights": "domaine public",
   "dc:title": "Projet de chemin de fer de Calais à Marseille rapport fait à
               la Chambre de commerce de Boulogne, le 25 novembre 1881",
   "dc:type": "text",
   "dc:source": "Bibliothèque Georges Lefebvre"
  }
 }
}
```


FIG. 2 – *Extrait JSON du modèle physique MongoDB de la collection* Sat_Book

compte des index, de l'espace mémoire non utilisé et de celui qui est libéré lors de la suppression ou du déplacement des données.

Nous constatons en premier lieu que le volume des métadonnées générées pour 245 fichiers est faible (moins d'1 Mo) dans les deux cas. Par ailleurs, MongoDB nécessite systématiquement moins d'espace que PostgreSQL. Cela est justifié par l'utilisation du format JSON, qui est très léger, dans MongoDB. En revanche, dans PostgreSQL, chaque table est stockée comme un vecteur de pages de taille prédéterminée (8 Ko), ce qui induit des pages non totalement remplies.

4.2 Temps de réponse

Dans le but de mesurer la performance du modèle de métadonnées proposé, nous avons formulé cinq requêtes de type projection/restriction et de complexité croissante en termes de nombre de centres impliqués (et donc de jointures via le lien $Link\_Document$), que nous appliquons sur les deux modèles physiques.

1. Récupérer tous les documents dont le titre contient le terme « industrielle » (sources de données : Hub_Title, Sat_Title).
2. Récupérer tous les documents dont le titre contient le terme « industrielle » et dont l'emplacement est « bibliothèque » (sources de données : Hub_Title, Sat_Title, $Hub\_ Location$, $Sat\_Location$, $Link\_Document$).
3. Récupérer tous les documents dont le titre contient le terme « industrielle », dont l'emplacement est « bibliothèque » et dont la date est « 2010 » (sources de données :

TAB. 4 – *Volume des métadonnées (Ko)*

	PostgreSQL	**MongoDB**
Hub_Title	80	45
Hub_Date	64	36
Hub_Location	72	49
Hub_Category	64	36
Link_Document	48	36
Sat_Title	168	69
Sat_Date	48	36
Sat_Location	56	49
Sat_IHRIS	16	16
Sat_VoixDuNord	16	7
Sat_Inventory	88	45
Sat_Book	56	37
Total	**776**	**461**

Hub_Title, Sat_Title, $Hub\_Location$, $Sat\_Location$, Hub_Date, Sat_Date, $Link\_Document$).

4. Récupérer tous les documents dont le titre contient le terme « industrielle », dont l'emplacement est « bibliothèque », dont la date est « 2010 » et qui appartiennent à la categorie *Ouvrage* (sources de données : Hub_Title, Sat_Title, $Hub\_Location$, $Sat\_Location$, Hub_Date, Sat_Date, $Hub\_Category$, Sat_Book, $Link\_Document$).
5. Récupérer tous les documents dont le titre contient le terme « industrielle » et qui appartiennent à une catégorie passée en paramètre (sources de données : Hub_Title, Sat_Title, $Hub\_Category$, Sat_IHRIS, $Sat\_VoixDuNord$, $Sat\_Inventory$, Sat_Book, $Link\_Document$).

La Figure 3 présente le temps d'exécution moyen des requêtes (1) à (4) sous PostgreSQL et MongoDB, respectivement. Nous avons exécuté 100 fois chaque requête (dans l'ordre indiqué ci-dessus) pour pallier les éventuelles variations de temps d'exécution. La Figure 3 montre des temps de réponse spectaculairement bas pour MongoDB, alors que ceux obtenus avec PostgreSQL semblent évoluer de manière exponentielle.

La Figure 4 illustre le temps d'exécution moyen de la requête (5), toujours pour 100 exécutions de la requête. Elle est distincte de la Figure 3 car la requête (5) est plus complexe que les quatre précédentes. En effet, il n'est pas possible de déterminer dynamiquement le satellite *Catégorie* à utiliser. Il faut donc exécuter la requête (5) en deux étapes (sous-requêtes) : la première pour déterminer la catégorie et la seconde pour récupérer le reste des informations sachant la catégorie. Ces deux opérations induisent des temps de réponse très supérieurs à ceux des requêtes (1) à (4).

MongoDB se montre une nouvelle fois plus efficace (trois fois plus rapide en moyenne) que PostgreSQL. Cette différence de temps d'exécution de la requête (5) s'explique par la réécriture en interne par PostgreSQL, qui crée une sous-requête induisant une jointure coûteuse (Giménez, 2011), alors que MongoDB utilise une méthode de recherche qui permet l'union rapide des collections.

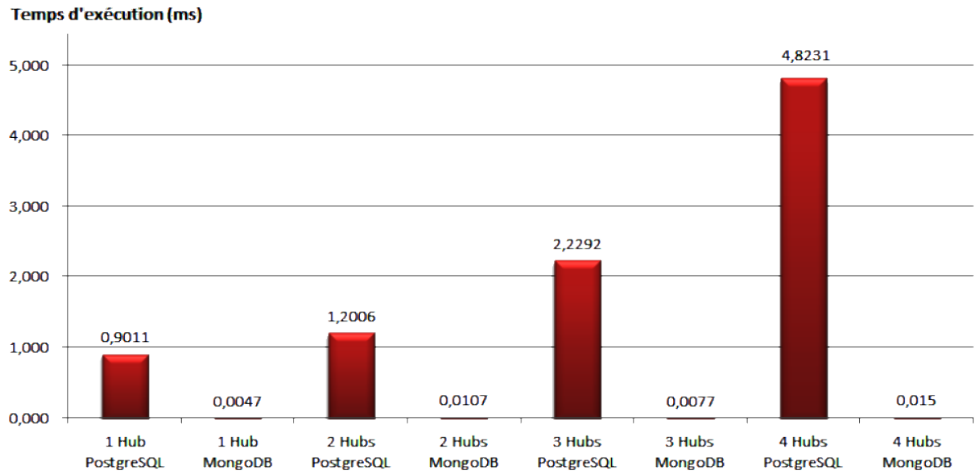

FIG. 3 – *Temps moyen de réponse des requêtes (1) à (4)*

5 Conclusion

Partant de la modélisation des métadonnées d'un lac de données sous forme multidimensionnelle et constatant que l'évolution du schéma n'était pas garantie, nous avons proposé dans cet article une modélisation ensembliste, et plus précisément en *data vault*, des métadonnées.

La traduction de notre modèle conceptuel de métadonnées en modèles logiques et physiques et des expériences menées sur le corpus TECTONIQ ont permis de montrer la faisabilité de notre choix de modélisation en termes de volume de stockage des métadonnées et de temps de réponse aux requêtes formulées sur les métadonnées.

La comparaison de deux modèles physiques (PostgreSQL et MongoDB) a également fait apparaître la supériorité des modèles logiques orientés document pour stocker ce type de métadonnées.

Les perspectives ouvertes par ce travail préliminaire sont nombreuses. Il conviendrait notamment de tester la robustesse du modèle de métadonnées lors d'un passage à l'échelle des données sources, d'ajouter des sources de données pour vérifier la pertinence de la modélisation en *data vault* et de tester des requêtes plus complexes que des projections/restrictions.

Il serait également intéressant de comparer l'efficacité de la modélisation en *data vault* et l'*anchor modeling*, les tenants de cette dernière argumentant que la modélisation en 6NF permet malgré tout de bons temps de réponse, grâce à la technique d'élimination de jointures employée dans les optimiseurs modernes (Paulley, 2000).

Finalement, différentes modélisations alternatives des métadonnées, indépendamment des techniques de modélisation employées, pourraient être envisagées et comparées. Les schémas des documents XML du corpus pourraient aussi être extraits automatiquement pour enrichir les métadonnées.

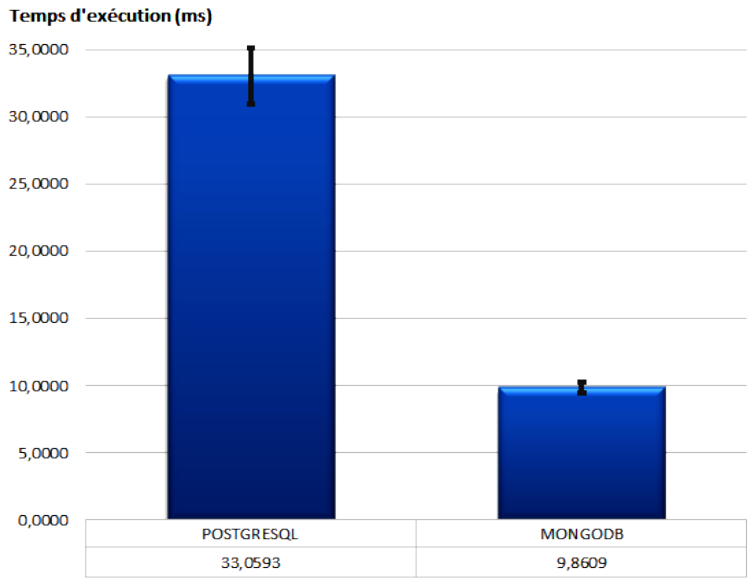

FIG. 4 – *Temps moyen de réponse de la requête (5)*

Remerciements

Les auteur·es remercient Eric Kergosien, porteur du projet TECTONIQ, pour la mise à disposition du corpus de données, ainsi que les évaluateur·trices de l'article pour leurs retours constructifs.

Références

Alrehamy, H. H. et C. Walker (2015). Personal Data Lake With Data Gravity Pull. In *IEEE 5th International Conference on Big Data and Cloud Computing (BDCloud 2015), Dalian, China*, pp. 160–167.

Clapeau, A. (2017). Qu'est-ce que le Data Lake, le nouveau concept "Big Data" en vogue. *Journal du Net*. http ://www.journaldunet.com/solutions/cloud-computing/1165409-qu-est-ce-que-le-datalake- le-nouveau-concept-big-data-en-vogue/.

Dixon, J. (2010). Pentaho, Hadoop and Data Lakes. https ://jamesdixon.wordpress.com/2010/10/14/pentaho-hadoop-and-data-lakes/.

Ganore, P. (2015). Introduction To The Concept Of Data Lake And Its Benefits. https ://www.esds.co.in/blog/introduction-to-the-concept-of-data-lake-and-its-benefits/.

Giménez, H. (2011). PostgreSQL Performance Considerations. https ://robots.thoughtbot.com/postgresql-performance-considerations.

Hultgren, H. (2012). Data vault modelling guide – Introductory guide to data vault modelling. *Genesee Academy*. https ://hanshultgren.files.wordpress.com/2012/09/data-vault-modeling-guide.pdf.

Jovanovic, V. et I. Bojicic (2012). Conceptual Data Vault Model. In *Southern Association for Information Systems Conference, Atlanta, GA, USA*, pp. 131–136.

Kergosien, E. (2017). TEchnologies de l'information et de la communication au Cœur du TerritOire NumérIQue pour la valorisation du patrimoine. https ://tectoniq.meshs.fr/.

Linstedt, D. (2011). *Super Charge your Data Warehouse : Invaluable Data Modeling Rules to Implement Your Data Vault*. CreateSpace Independent Publishing.

Linstedt, D. (2015). Data Vault Basics. https ://danlinstedt.com/solutions-2/data-vault-basics/.

Linstedt, D. et M. Olschimke (2015). *Building a Scalable Data Warehouse with Data Vault 2.0*. Morgan Kaufmann.

Miloslavskaya, N. et A. Tolstoy (2016). Big Data, Fast Data and Data Lake Concepts. *Procedia Computer Science 88*, 300–305.

Pathirana, N. (2015). Modeling territorial knowledge from web data about natural and cultural heritage. Mémoire de Master, Université Lumière Lyon 2.

Paulley, G. N. (2000). *Exploiting Functional Dependence in Query Optimization*. Ph. D. thesis, University of Waterloo, Canada.

Regardt, O., L. Rönnbäck, M. Bergholtz, P. Johannesson, et P. Wohed (2009). Anchor Modeling. In *28th International Conference on Conceptual Modeling (ER 2009), Gramado, Brazil*, Volume 5829 of *Lecture Notes in Computer Science*, pp. 234–250.

Rönnbäck, L. et H. Hultgren (2013). Comparing Anchor Modeling with Data Vault Modeling. https ://hanshultgren.files.wordpress.com/2013/06/modeling_compare_05_larshans.pdf.

Rönnbäck, L., O. Regardt, M. Bergholtz, P. Johannesson, et P. Wohed (2010). Anchor modeling – Agile information modeling in evolving data environments. *Data and Knowledge Engineering 69*(12), 1229–1253.

Stein, B. et A. Morrison (2014). The enterprise data lake : Better integration and deeper analytics. *Technology Forecast*, 1. http ://www.pwc.com/us/en/technology-forecast/2014/cloud-computing/assets/pdf/pwc-technology-forecast-data-lakes.pdf.

Summary

With the rise of big data, business intelligence had to find solutions for managing even greater data volume and variety than in data warehouses, which proved ill-adapted. Data lakes answer these needs from a storage point of view, but require managing adequate metadata to guarantee an efficient access to data. Starting from a multidimensional metadata model designed for an industrial heritage data lake presenting a lack of schema evolutivity, we propose in this paper to use ensemble modeling, and more precisely a data vault, to address this issue. To illustrate the feasibility of this approach, we instantiate our metadata conceptual model into relational and document oriented logical and physical models, respectively. We also compare the physical models in terms of metadata storage and query response time.