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, l'informatique décisionnelle a dû trouver des solutions pour gérer des données de très grands volume et variété. 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 présentant un défaut d'évolutivité de schéma, nous proposons l'utilisation 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. Nous comparons également les modèles physiques en termes de stockage et de temps de réponse aux requêtes sur les métadonnées.

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, en vue de les analyser (*reporting*, visualisation, fouille de données...).Ce concept s'oppose à celui des entrepôts de données, très intégrés et orientés sujet, mais qui ont 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. Ce modèle multidimensionnel est proche des modèles en flocons en usage dans les entrepôts de données, mais ne concerne que les métadonnées et non le corpus de documents lui-même. Il a été instancié au niveau physique dans différents systèmes de gestion de bases de données (SGBD) NoSQL. 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), 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.

2 État de l'art

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, 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).

La modélisation ensembliste (*ensemble modeling*) est une approche utilisée dans l'industrie, qui vise à renormaliser les entrepôts de données afin de permettre une meilleure évolutivité, tant en termes de données que de schéma (Rönnbäck et Hultgren, 2013). Les deux approches qui s'en dégagent sont l'*anchor modeling* (Regardt et al., 2009) et les *data vaults* (Linstedt, 2011). 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 et sont supportés par un plus grand nombre d'outils, ce qui a guidé notre choix.

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). Plus concrètement, un *data vault* est composé des éléments principaux suivants. 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 des attributs relatifs à un centre ou un lien.

3 Modèle de métadonnées en *data vault*

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 : descriptions de bâtiments industriels (documents XML orientés données), articles de presse liés à l'industrie textile (documents XML orientés documents), photos et plans de bâtiments et monuments liés à l'industrie textile (images JPEG) et livres d'histoire de France du domaine public (documents PDF). Il 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.

La Figure 1 illustre notre modèle conceptuel de métadonnées en *data vault*. Les rectangles arrondis bleus y représentent les centres, les rectangles gris les satellites et l'hexagone vert un lien. 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 ».

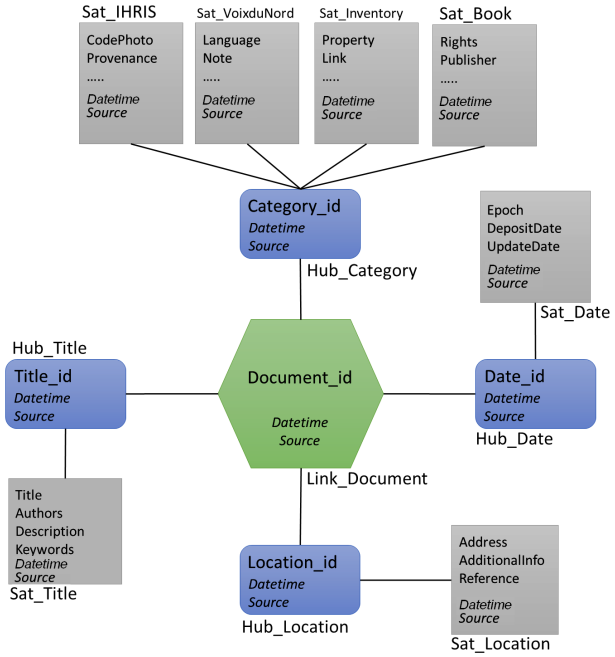


FIG. 1: Modèle de métadonnées en *data vault*

Afin d'identifier les centres, 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. 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. 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*).

Pour montrer que notre modèle conceptuel peut s'adapter à différents contextes, et pour 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 est traduit du modèle conceptuel (Figure 1) de manière classique. Notons seulement que, dans les satellites, la clé primaire est constituée de la clé primaire du centre dont dépend le satellite et de l'attribut *Datetime*.Le modèle orienté document des métadonnées exploite la notion de collection. Chaque centre, lien et satellite est traduit en collection comportant des documents ayant chacun un identifiant (*ObjectID*). Dans le modèle physique de MongoDB, cet identifiant est généré automatiquement à la création d'un document. Chaque centre est associé au lien *Link_Document* et à son ou ses satellites grâce à son *ObjectID*.

4 Validation expérimentale

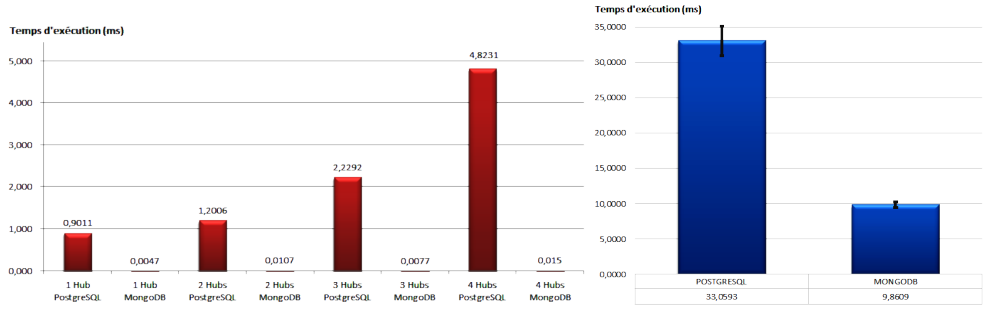
L'objectif de cette section est de montrer la faisabilité de notre modèle de métadonnées en *data vault* et de comparer les modèles physiques PosgreSQL et MongoDB. Nous avons mené nos expériences sur un PC Intel Core i5-5300U 2,30 GHz avec 8 Go de mémoire, sous Windows 64 bits. Les versions des SGBD sont PostgreSQL 9.6.2-1 et MongoDB ssl 3.4.3.

Après avoir inséré les métadonnées dans le format natif des SGBD choisis via des scripts, nous avons mesuré le volume de stockage nécessaire. Nos mesures tiennent compte des index, de l'espace mémoire non utilisé et de celui qui est libéré lors de la suppression ou du déplacement des données. 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.

Dans le but de mesurer la performance de notre modèle de métadonnées, 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) Documents dont le titre contient le terme « industrielle ». (2) Documents dont le titre contient le terme « industrielle » et dont l'emplacement est « bibliothèque ». (3) Documents dont le titre contient le terme « industrielle », dont l'emplacement est « bibliothèque » et dont la date est « 2010 ». (4) 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*. (5) Documents dont le titre contient le terme « industrielle » et qui appartiennent à une catégorie passée en paramètre.

La Figure 2a 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 in-

diqué ci-dessus) pour pallier les éventuelles variations de temps d'exécution. La Figure 2a montre des temps de réponse spectaculairement bas pour MongoDB, alors que ceux obtenus avec PostgreSQL semblent évoluer de manière exponentielle. La Figure 2b illustre le temps d'exécution moyen de la requête (5) sur 100 exécutions. Elle est distincte de la Figure 2a car la requête (5) est plus complexe que les 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 : 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 s'explique par la réécriture en interne par PostgreSQL d'une sous-requête induisant une jointure coûteuse, alors que MongoDB permet l'union rapide des collections.



(a) Requêtes (1) à (4) (b) Requête (5)

FIG. 2: Temps moyen de réponse des requêtes

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 une modélisation 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 approche en termes de volume de stockage 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 incluent 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. 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.

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.

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.

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

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.

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.

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.

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 devised solutions for managing great data volume and variety. Data lakes are an answer from a storage point of view, but require managing adequate metadata to guarantee efficient data access. From a multidimensional metadata model designed for a data lake presenting a lack of schema evolutivity, we propose to use 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. We also compare the physical models in terms of metadata storage and query response time.