Évaluation et comparaison de structures de données orientées document

Evaluation and comparison of document-oriented data structures

Paola Gómez[1], Claudia Roncancio[1], and Rubby Casallas[2]

[1]Univ. Grenoble Alpes, CNRS, Grenoble INP *, LIG, 38000 Grenoble, France
{paola.gomez-barreto,claudia.roncancio}@univ-grenoble-alpes.fr
[2]TICSw, Universidad de los Andes, Bogotá - Colombia
rcasalla@uniandes.edu.co

RÉSUMÉ. La flexibilité dans la structuration des données dans les bases orientées document est appréciée pour permettre un développement initial rapide. Cependant, les possibilités de structuration des données sont nombreuses et le choix de structuration adopté reste assez crucial par son impact potentiel sur plusieurs aspects de la qualité des applications. En effet, chaque structuration peut présenter des avantages et des inconvénients notamment en matière d'empreinte mémoire, redondance de données engendrée, coût de navigation dans les structures et accès à certaines données. Cet article introduit le projet SCORUS qui vise à assister les utilisateurs dans la modélisation des données pour des bases orientées document. La principale contribution est la proposition d'un ensemble de métriques structurelles pour des "schémas" de documents JSON. Ces métriques permettent de refléter la complexité des schémas et des critères de qualité tels que leur lisibilité et maintenabilité. La définition des métriques est complétée par un scénario de validation.

ABSTRACT. Document oriented bases allow high flexibility in data representation. This facilitates a rapid development of applications and allows many possibilities for data structuration. Nevertheless, the structuration choices remain crucial because they impact several aspects of the document base and application quality, e.g, memory print, data redundancy, querying and navigation facility. It is therefore important to be able to analyse and to compare several data structuration alternatives. We introduce in this paper our SCORUS project, which aims to assist users in the modeling of data for document-oriented databases. The main contribution is the proposal of a set of structural metrics structural for JSON documents. They mesure several aspects of the complexity of the structure in order to be used in criteria helping in the schema design process. This paper presents the definition of the metrics together with a validation scenario.

MOTS-CLÉS. NoSQL, métriques structurelles, systèmes orientés document, MongoDB
KEYWORDS. NoSQL, structural metrics, document-oriented systems, MongoDB

1. Introduction

L'essor des systèmes NoSQL nous amène à réfléchir à leur positionnement dans les systèmes d'information dans la durée. Nous nous intéressons notamment à des questions de qualité liées à la structuration des données au sein de systèmes orientés document, type MongoDB. Dans ce système, comme dans la plupart des systèmes NoSQL, le système de types est riche et il n'y a pas de gestion explicite d'un schéma de base de données. Cela octroie une grande flexibilité à l'utilisateur tout en introduisant une complexité dans l'accès aux données. La manière de structurer les données est importante car elle a un impact sur divers aspects de la base de documents et des applications qui les utilisent (Gómez *et al.*, 2016 ; Gomez, 2018). Les choix de structuration peuvent présenter des avantages et des inconvénients en matière d'empreinte mémoire, redondance de données engendrée, coût de navigation dans les structures et d'obtention de certaines données, et lisibilité des programmes.

Nos travaux visent à aider l'utilisateur à comprendre, évaluer, choisir, et, potentiellement, faire évoluer la structuration des données d'une base orientée document (style JSON). Il s'agit de permettre de

* Institute of Engineering Univ. Grenoble Alpes

considérer plusieurs structurations candidates pour retenir un choix unique, ou temporel, ou plusieurs alternatives parallèles selon les cas.

Ainsi, nous avons fait le choix d'expliciter le type des structures de données utilisées dans la base. Pour cela nous proposons le format AJSchéma, simple et lisible. Sans jouer le rôle de schéma dans la base de documents, un AJSchéma permettra néanmoins de mettre en évidence les types des données et ainsi évaluer et raisonner sur les caractéristiques d'une structure. L'analyse de ses caractéristiques croisées avec les préférences des utilisateurs permettra de mettre en avant les structures à privilégier ou à écarter. Nos contributions sont coordonnées dans une approche appelée SCORUS qui distingue des phases de modélisation, de génération, d'évaluation et d'analyse de structurations de données.

L'élément central de cet article [1] est la présentation d'un ensemble de métriques structurelles pour des documents JSON. Ces métriques permettent de refléter la complexité de la structure et peuvent être utilisées pour établir des critères de qualité tels que leur lisibilité et maintenabilité. La définition de ces métriques s'appuie, entre autres, sur des travaux sur MongoDB, XML et en Génie logiciel pour la qualité du code.

SCORUS permet de générer automatiquement un ensemble de variantes de structures orientés document et de les évaluer à l'aide des métriques proposées dans cet article. Il s'agit de faciliter leur analyse afin d'aider les développeurs à faire des choix éclairés.

Dans la suite, la section 2 rappelle certains éléments de MongoDB et la motivation de nos travaux. La section 3 propose une vue globale de SCORUS. Ensuite la section 4 se concentre sur l'étape d'évaluation des alternatives de structuration et propose un ensemble de métriques permettant de les comparer. La section 5 fournit un scénario de validation qu'utilise les métriques pour comparer des structurations de données. Les travaux connexes sont décrits en section 6. Nos conclusions et perspectives de recherche sont présentées en section 7.

2. Contexte et Motivation

Comme déjà mentionné, nous nous intéressons à des questions de qualité de structuration de données JSON au sein de systèmes type MongoDB. Rappelons que dans ce système (comme dans la plupart des NoSQL), il n'y a pas de gestion explicite d'un schéma de données. La manière de structurer les données reste néanmoins importante car elle a un impact sur plusieurs aspects de la base de documents et des applications qui les utilisent.

Le format BSON gère les données comme un ensemble de collections de documents (cf. Figure 2a, collections *Companies* et *Departments*). Un document est simplement un ensemble de paires `attribut : attribut : valeur`. Le type des valeurs peut être atomique ou complexe. Par complexe, nous entendons soit un tableau de valeurs de tout type ou un document qui dans ce cas est dit *imbriqué* (cf. Figure 2b). Notons que la valeur d'un attribut peut être l'identifiant d'un document ou la valeur d'un attribut d'un document d'une autre collection. Cela permet de *référencer* un ou plusieurs documents.

1. Cet article est une version étendue de (Gómez *et al.*, 2018). Il augmente ce dernier en présentant une version plus complète de la définition des métriques et une introduction de SCORUS.

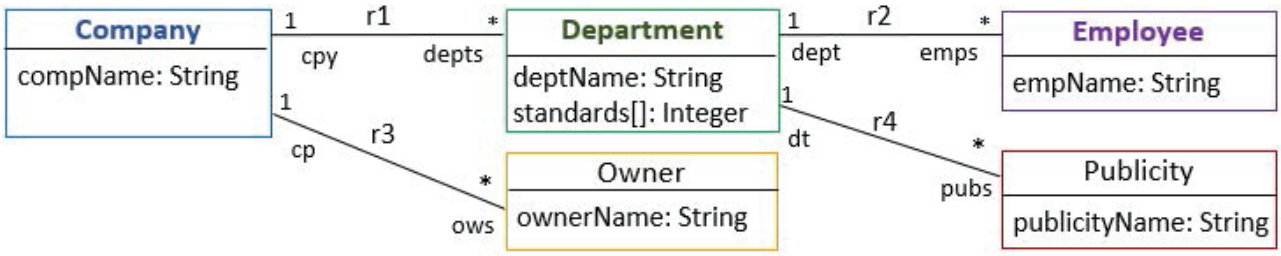


Figure 1. *Diagramme UML dans un contexte de Marketing*

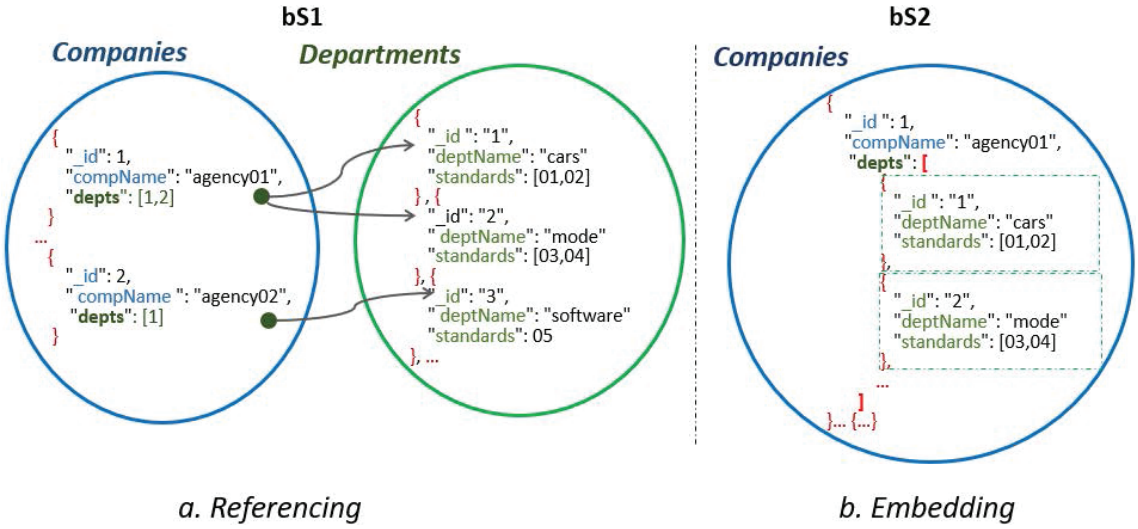


Figure 2. *Exemples d'instances MongoDB pour des données modélisées sur la Figure 1 (Classes* `Company` *et* `Department`*)*

Ce système de types est à la fois simple et puissant car il permet beaucoup de flexibilité dans la création de structures complexes. Dans un cas simple comme celui de notre exemple, l'association 1-N entre `Company` et `Department` du modèle UML de la figure 1, est susceptible d'être représentée de plusieurs manières. La figure 2a montre un choix pour la collection *Companies* qui utilise des références vers *Departments* alors que le choix illustré sur 2b imbrique les documents correspondant aux "departments". Sur la Figure 2b la collection *Departments* n'est pas créée et il n'y a pas de duplication de données.

De manière générale, tout en garantissant la complètude des données, à partir d'un modèle UML par exemple, il est possible de construire plusieurs structurations orientées document, e.g. collections séparées et sans imbrication, collections complètement imbriquées, combinaison d'imbrications et de référencements ou duplication de données. Le choix de la meilleure structuration n'aura probablement pas une réponse unique, ni absolue, et dépendra des priorités et besoins d'accès du moment.

La manière dont sont structurées les données a un fort impact sur la taille de la base, les performances des requêtes et la lisibilité du code des requêtes, ce influence la maintenabilité et l'usabilité de la base ainsi que de ses applications. Cela a été constaté de manière expérimentale (Gómez *et al.*, 2016) où plusieurs patrons de comportement ressortent. Notamment, les collections avec des documents imbriqués ont un impact positif sur les requêtes qui suivent l'ordre d'imbrication. Cependant, il n'y a aucun avantage — ou il y a la possibilité de mauvaises performances — pour les requêtes qui accèdent aux données dans un autre ordre d'imbrication ou à des données imbriquées à différents niveaux dans la même collection. Les mauvaises performances sont liées à la nécessité de manipulations d'une complexité similaire à celles

de jointures avec plusieurs collections. En outre, les collections avec des documents imbriqués — même sans duplication — ont tendance à exiger plus de stockage que les mêmes données représentées sur les collections distinctes.

Dans le choix de la structuration des données, les priorités peuvent s'avérer divergentes, comme le souhait de dupliquer des documents dans plusieurs collections parce qu'ils sont consultés dans des contextes différents mais aussi le souhait de réduire le coût du stockage. Nous cherchons, d'une part, à clarifier l'impact de la structuration de données sur la qualité des applications et, d'autre part, à proposer des aides aux développeurs pour faire des choix éclairés au sujet de la structuration des données.

Pour assister l'utilisateur dans l'évaluation et le choix des structures, nous proposons SCORUS dont les principes généraux et la génération des alternatives sont introduits dans la section suivante. Ensuite, dans la section 4, nous approfondissons nos propositions en matière de métriques. Elles constituent des éléments objectifs d'appréciation et de comparaison. L'analyse de ces éléments croisés avec les préférences des utilisateurs permettra de mettre en avant les structures à privilégier ou à écarter, comme nous l'apprécierons dans notre scénario de validation.

3. *SCORUS*, un système pour l'analyse et l'évaluation des structures orientés document

Cette section introduit SCORUS [2], un système pour l'analyse et l'évaluation de structures orientées document. La section 3.1 présente l'approche globale et ses phases. La section 3.2 donne les principaux choix pour la phase de génération d'alternatives de structuration. Nos propositions sur les métriques structurelles sont détaillées dans la section 4.

3.1. *Approche globale*

La flexibilité et l'absence de schéma dans les systèmes NoSQL orientés document, tels que MongoDB, permettent d'explorer de nouvelles alternatives de structuration sans faire face aux contraintes. Le choix de la structuration reste important et critique parce qu'il détermine divers aspects de la base et des programmes qui l'utilisent. De plus de nombreuses options de structuration sont possibles. Nous proposons d'adopter une phase de conception dans laquelle des aspects de qualité et les impacts de la structure sont pris en compte afin de prendre une décision de structuration des documents d'une manière plus avertie.

Dans ce cadre, *SCORUS* vise à faciliter l'étude des possibilités de structuration orientée document et à fournir des métriques objectives pour mieux faire ressortir les avantages et les inconvénients de chaque solution par rapport aux besoins des utilisateurs. Pour cela, une séquence de trois phases peut composer un processus de conception (illustré par la Figure 3) ou être effectuées indépendamment à des fins d'analyse et de réglage. Les phases sont :

— *Génération d'un ensemble d'alternatives de structuration* : dans cette phase (cf. section 3.2) nous proposons de partir d'une modélisation UML des données, de produire automatiquement un large ensemble de variantes de structuration possibles selon l'approche orientée document.

— *Evaluation d'alternatives en utilisant un ensemble de métriques structurelles* : cette évaluation porte sur une variante ou un ensemble de variantes de structuration et calcule les métriques qui

2. Semi Structured SChemas Recommendation System

reflètent les principales caractéristiques au regard des données modélisées. Les métriques proposées sont présentées dans la section 4.

— *Analyse des alternatives évaluées* : il s'agit d'utiliser les évaluations des métriques de chaque alternative de structuration pour les analyser et les comparer afin de choisir la ou les plus appropriées. Pour cela, des critères peuvent être établis selon le contexte applicatif mais peuvent aussi correspondre à de bonnes pratiques préconisées pour le développement ou à une priorité générale. Dans le cadre de cette article nous présentons une analyse non automatisée qui sera introduite dans la section 5, consacrée à la validation.

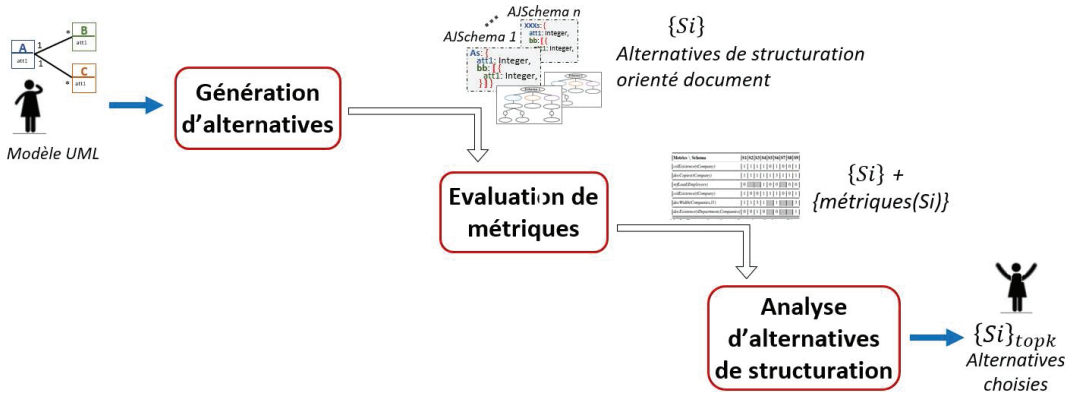


Figure 3. *Organisation générale de SCORUS*

Pour faciliter le raisonnement sur les structures de données, nous proposons de mettre l'accent sur les types des données contenues dans les documents. Les types de données peuvent être représentés sous forme de structures de documents qui pourraient être assimilés à des éléments d'un schéma d'une base. Pour la réalisation de SCORUS, nous manipulons deux représentations :

1. une représentation arborescente, nommée *AJTree*. Cette représentation est principalement destinée à l'évaluation automatique des métriques. La Figure 6 montre un exemple utilisé lors de l'introduction des métriques en section 4.
2. une représentation textuelle, nommée *AJSchéma*. Elle est basée sur l'approche JSON schema. Elle est centrée sur une présentation finale claire des structures. La Figure 5 montre des exemples d'alternatives de structures.

3.2. *Génération d'alternatives de structuration*

Pour réaliser la phase de génération, *SCORUS* part d'un modèle UML et produit un ensemble de variantes d'AJSchémas pouvant le représenter. Cette génération automatique permet à l'utilisateur d'apprécier facilement de multiples choix de structures dont l'analyse peut être poursuivie ou non. Pour ce générateur nous avons suivi une approche issue du domaine des lignes de produits logiciels. Il s'agit de l'approche par modèles de caractéristiques qui permet d'exprimer la variabilité entre diverses alternatives d'un produit (Kang *et al.*, 2002) (Gomaa, 2005). Ici, nous cherchons à produire diverses alternatives d'AJSchéma.

Nous utilisons les éléments du modèle UML et les possibilités de structuration orientées document afin d'identifier les alternatives possibles. Pour cela nous avons formalisé les variations et points communs à travers un modèle de caractéristiques. La stratégie des modèles de caractéristiques permet de définir la

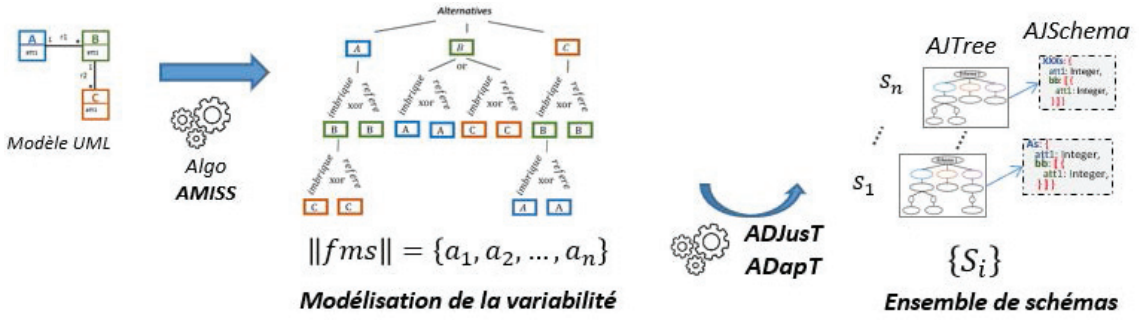


Figure 4. *Phase de génération d'alternatives*

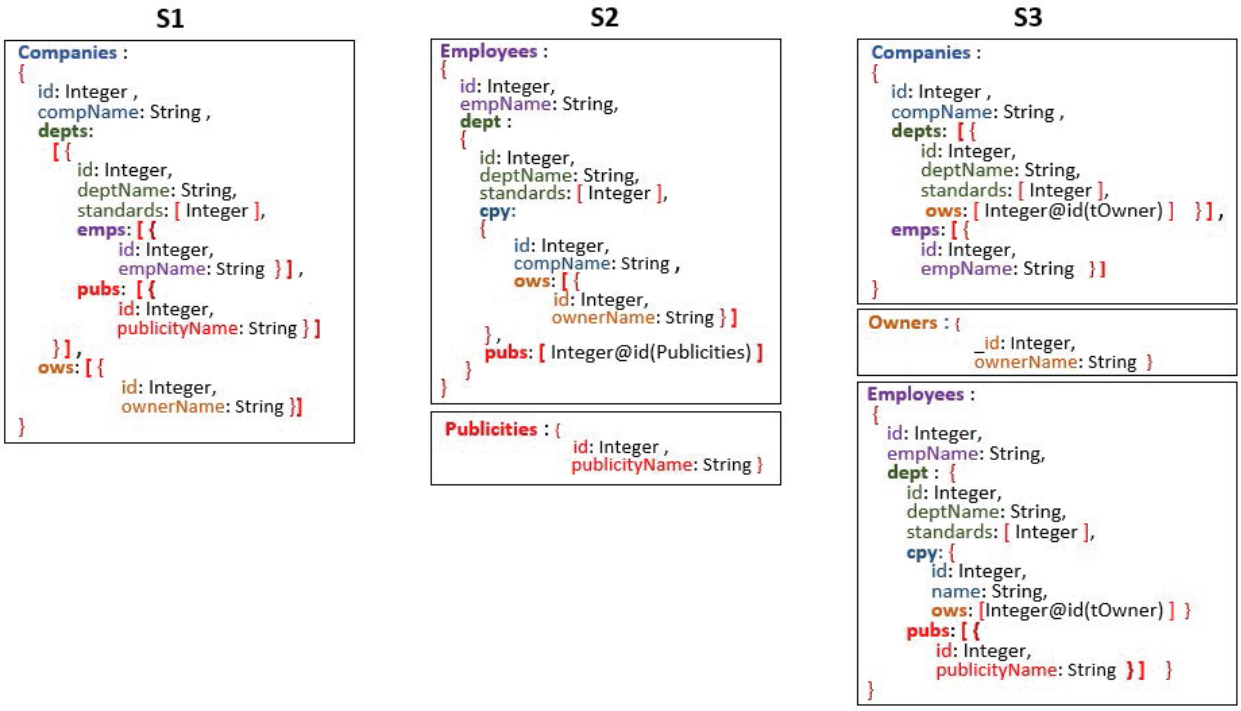


Figure 5. *Trois alternatives de structuration, sous forme AJSchéma, pour le modèle UML de la Figure 1*

variabilité des structures possibles d'une manière compacte, et de contrôler l'explosion des possibilités qui peuvent survenir. Nous avons développé l'algorithme *AMISS* (Gómez *et al.*, 2019) qui pour un modèle UML [3] crée un modèle de caractéristiques contenant les variations de structuration.

Une modélisation du *cas simple* (deux classes et une association) permet de définir des choix de modélisation tels que : la création d'une collection pour chacune des deux classes et l'association, ses types de documents et les alternatives qui modélisent soit l'imbrication, soit le référencement de documents correspondant à la classe opposée.

AMISS utilise la modélisation du cas simple afin de modéliser *plusieurs classes et associations*. Un parcours du modèle UML crée de manière incrémentale le modèle de caractéristiques complet, nommé fm_s. Ce modèle représente toutes les alternatives de structuration proposées pour une base orientée documents. Le modèle fm_s inclut une collection par classe et par association (pour les éventuelles colletions-lien). Pour chaque collection de classe, le type des éléments inclut les attributs propres à la classe et des

3. Cet article se limite aux associations binaires sans attributs

attributs supplémentaires possibles pour représenter les associations de la classe soit par imbrication, soit par référencement de documents.

Pour chaque alternative de structuration fournie par ce modèle, le système dérive les structures orientées document correspondantes à donner en sortie. Ces structures peuvent être visualisées dans divers formats dont les AJSchémas ou sous forme arborescente (AJTree). Cette dernière représentation sert notamment lors de l'évaluation automatique des métriques.

Nous introduisons ci-après un exemple de la dérivation vers les AJSchemas (cf. Figure 5) et ensuite nous le détaillons sur l'AJTree pour des explications complémentaires sur cette phase de génération.

Faute de place, la phase de génération ne sera pas détaillée dans cet article. Le lecteur intéressé peut se référer à (Gómez *et al.*, 2019) et (Gomez, 2018).

Example 3.1. Pour l'exemple, considérons la Figure 1 qui développe un modèle de classes UML dans un contexte de Marketing.

La classe `Companpy` représente une compagnie ou agence qui gère plusieurs départements de publicité, représentés par la classe `Department`. Chaque département est en charge de plusieurs employés (classe `Employee`) et est lié à plusieurs publicités (classe `Publicity`). Chaque employé et publicité appartient à un seul département et un département est géré par une seule compagnie. Chaque classe fournit son type pour indiquer ses attributs : $tCompany$, $tDepartment$, $tOwner$, $tEmployee$, $tPublicity$.

Ces données peuvent être représentées de nombreuses manières dans une base de documents. Le générateur va produire un ensemble d'alternatives qui peut ensuite être analysé. La Figure 5 introduit trois alternatives de structuration sous forme d'AJSchema qui diffèrent dans leur choix de collections et de structure des documents. L'alternative $S2$ contient deux collections. La collection `Publicities` dont les documents sont constitués par des attributs correspondant au type $tPublicity$. La deuxième collection, `Employees`, contient des documents avec des attributs de type $tEmployee$ et un attribut supplémentaire qui imbrique les attributs de `Department`. Ce dernier imbrique un document, nommé `cpy`, avec les attributs de $tCompany$ et référence les documents de la collection `Publicities`.

Pour ce cas le générateur propose une centaine d'alternatives. Ce nombre croît de manière exponentielle avec le nombre d'associations. Notre préconisation est de restreindre les alternatives en introduisant des contraintes qui représentent, entre autres, des préférences de structuration données par l'utilisateur. La formalisation des éléments permettant d'analyser le nombre d'alternatives reste un sujet à approfondir.

3.3. *Structure de graphes et exemple*

Afin de faciliter l'évaluation automatique d'une alternative de structuration à l'aide de nos métriques, nous proposons également une représentation arborescente, nommée AJTree, axée sur les types d'informations, les niveaux, l'imbrication et les références.

Nous considérons un modèle de données UML, avec un ensemble de classes $E = \{e_1, ..., e_n\}$. Les propriétés ou attributs d'une classe e_i sont désignés par le type te_i et ses associations par l'ensemble $R(e_i) = \{r_1, ..., r_n\}$. Pour chaque association, on connaît le rôle des entités reliées. A partir d'un même modèle plusieurs structurations orientées document sont envisageables. Le graphe de la figure 6 illustre la structure arborescente correspondant à l'AJSchéma S3 de la figure 5.

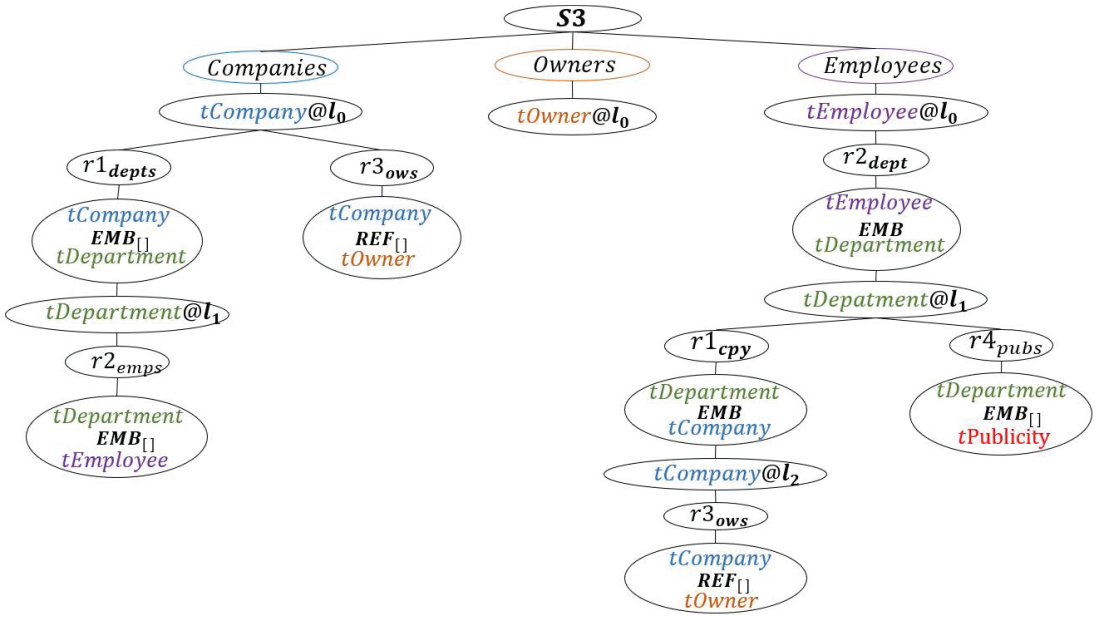


Figure 6. *Exemple de représentation des graphes de l'AJSchema S3 de la figure 5*

Le nœud racine, S3, a un nœud fils par *collection* présente (e.g . collections *Companies, Employees, Owners*). Il s'agit de collections de documents dont le type est représenté par le sous-arbre fils, nœud avec le suffixe l_0 (e.g. $tCompany@l_0$ pour documents company au niveau 0). Dans ce sous-arbre, les attributs de type atomique sont dans ce nœud $@l_0$ [4], et les attributs complexes (imbrication ou référence de documents) sont exprimés dans les nœuds fils. Les références sont matérialisées par des *nœuds REF* et les imbrications de documents par des *nœuds EMB* (e.g. $tCompanyEMB[]tDepartment$).

Lors de la création des structures orientées document, les attributs complexes sont utilisés pour stocker les associations $R(e_i)$ des classes e_i. Un nœud avec le nom de l'association est créé (e.g. $r2_{dept}$) avec pour fils un nœud *REF* ou *EMP* selon le choix. Des tableaux, notés $[]$, peuvent être utilisés pour les association n-aires. Par exemple, pour les départements d'une compagnie, association $r1_{depts}$, le document company aura un attribut de type tableau de documents business (nœud $tCompanyEMB[]tDepartment$).

Les imbrications de documents induisent un niveau de profondeur supplémentaire dans la structure. Ceci est matérialisé dans l'arbre par un *nœud niveau* l_i qui permet de savoir facilement à quel niveau de profondeur se trouve un document (e.g. $tDepartment@l_1$). Concernant les références, elles apparaissent à différents niveaux et référencent le type apparaissant au niveau 0 d'une collection.

4. Évaluation de métriques structurelles

Dans le choix de la structuration des données, les priorités peuvent s'avérer divergentes et il devient alors intéressant de pouvoir considérer plusieurs structurations candidates pour retenir un seul choix ou plusieurs alternatives parallèles pour un certain temps selon les cas. Pour faciliter l'étude d'alternatives nous avons exposé dans la section précédente les principes généraux de SCORUS et nous avons proposé une stratégie de génération de structures. Maintenant, nous abordons la question de l'absence d'indicateurs objectifs qui permettent d'aider dans le choix d'une structuration.

4. Attributs non listés sur la figure

Catégorie	Nom des métriques	Description	Par		
			sch	Col	type
Existence	*colExistence*	Existence d'une collection		x	
	docExistence	Existence d'un type de document dans une collection		x	x
	nbrCol	Nombre de collections	x		
Imbrication	*colDepth*	Profondeur maximale d'une collection		x	
	globalDepth	Profondeur maximale d'un schéma	x		
	DocDepthInCol	Niveau où un type de document se trouve dans une collection		x	x
	maxDocDepth	Niveau le plus profond où apparaît un type de document	x		
	minDocDepth	Niveau le mois profond où apparaît un type de document	x		
Largeur	*docWidth*	"Largeur" d'un type du document		x	x
	nbrAtomicAttributes	nombre d'attributs de type atomique			x
	nbrDocAttributes	nombre d'attributs de type document			x
	nbrArrayAtomicAttributes	nombre d'attributs de type tableau de valeurs atomiques			x
	nbrArrayDocAttributes	nombre d'attributs de type tableau de documents			x
Référencement	*refLoad*	Nombre de références à une collection		x	
Redondance	*docCopiesInCol*	Copies d'un type de document t dans une collection		x	x
	docTypeCopies	Nombre d'utilisations d'un type de document	x		

Tableau 1. *Métriques structurelles proposées*

Dans cette section nous proposons un ensemble de métriques structurelles, utilisées dans la deuxième étape de SCORUS, qui reflètent des aspects clés de la complexité des "schémas" semi-structurés. L'objectif est de faciliter plus tard leur analyse et comparaison indépendamment des instances. Les métriques peuvent être utilisées sans créer la base de données mais des informations statistiques sur les données peuvent bien sûr compléter l'analyse lorsqu'elles sont disponibles. Le tableau 1 résume les métriques qui seront définies dans la suite (sections 4.1 à 4.5).

Ce travail fait suite aux expériences sur des bases MongoDB opérationnelles (Gómez *et al.*, 2016). Nous avons fait une analyse du point de vue structurel afin de déterminer l'impact de la structuration et établir les caractéristiques qui sont en relation. Cela a conduit à la proposition des métriques présentées ici qui isolent divers aspects de la complexité des structures. Cette proposition s'appuie sur des travaux connexes (cf. section 6) mais est nouvelle en tant qu'outil quantitatif pour l'évaluation automatique de structures orientées document.

Afin de faciliter la manipulation automatique des variantes de structures nous utilisons la représentation AJTree, introduite brièvement dans la section 3.3 avec l'exemple fil rouge utilisé lors de la définition des métriques.

4.1. *Métriques d'existence*

Le choix de créer une collection pour un type de document sera motivé principalement par le besoin d'accès rapide ou fréquent à l'extension du type ou à un document du type en question. Au contraire, l'imbrication d'un type du document dans un autre peut être motivé par le fait que l'information est fréquemment consultée conjointement. S'assurer qu'un type de document n'est pas imbriqué à certains endroits peut aussi être intéressant, notamment si le document est peu accédé dans ce contexte ou si l'on cherche à réduire la complexité d'une collection. Dans cette catégorie nous définissons des métriques qui reflètent l'existence d'un type de document t dans un schéma. Nous considérons deux cas : (1) l'existence

d'une collection de documents de type t, et (2) la présence de documents de type t imbriqués dans d'autres documents. Ces cas sont couverts respectivement, par la métrique *colExistence* et la métrique *docExistence* définies ci-après.

Existence de collection : permet d'identifier si les collections ont des nœuds fils au niveau 0 correspondant au type t.

$$colExistence(t) = \begin{cases} 1 & \text{le nœud } t@l_0 \text{ apparaît dans le schéma} \\ 0 \end{cases} \quad (1)$$

Existence de type imbriqué : l'imbrication de documents de type t est matérialisé dans le graphe par un nœud $*EMB\,t$.

$$docExistence(\varphi, t) = \begin{cases} 1 & t \in \varphi \quad \text{un nœud } *EMB\,t \text{ apparaît dans un chemin partant de } \varphi \\ 0 & t \notin \varphi \end{cases} \quad (2)$$

Notons sur la figure 6 que pour les types *tCompany, tOwner et tEmployee* il y a des collections (nœuds $@l_0$) alors que ce n'est pas le cas pour *tPublicity*. Ceux-ci existent exclusivement imbriqués dans des documents *tEmployee*. Notons aussi que des documents de type *tDepartment* sont imbriqués dans deux collections, *Companies et Employees*. Nous verrons dans la suite que la prise en compte de ce fait peut s'avérer pertinent dans l'analyse des schémas.

Nombre de collections : son évaluation est le nombre de nœuds enfants de la racine du graphe.

$$nbrCol() = n \quad \text{nombre de nœuds enfants de la racine} \quad (3)$$

Sur la Figure 6, le schéma contient trois collections représentées par les nœuds enfants de la racine (*Companies, Owners* et *Employees*).

4.2. *Métriques d'imbrication*

En général, plus une information est imbriquée profondément, plus il est coûteux d'y accéder sauf si l'information intermédiaire est aussi recherchée par la requête. Savoir à quel niveau d'imbrication apparaît un type de document permet d'évaluer les coûts de navigation et d'aller-retour entre les niveaux ("intra-joint") pour y accéder, ou si des opérations de restructuration sont nécessaires pour extraire le format le plus approprié. Cette catégorie est consacrée aux métriques qui indiquent le niveau d'imbrication des documents.

Profondeur d'une collection : la métrique *colDepth* (4) indique le niveau de profondeur où se trouve le document le plus imbriqué. L'imbrication des documents est représentée par les nœuds EMB dans le graphe.

$$colDepth(\varphi) = max(depth(p_i)) \quad p_i \text{ est un chemin partant du nœud } \varphi \quad (4)$$

$$depth(p) = n \quad \text{nombre de nœuds } EMB \text{ dans le chemin } p \quad (5)$$

Profondeur d'un schéma : la métrique *globalDepth* (6) indique le niveau d'imbrication le plus profond des collections d'un schéma.

$$globalDepth(x) = max(colDepth(\varphi_i)) \qquad \forall \text{ collection } \varphi_i \in x \tag{6}$$

Connaître la profondeur d'imbrications des collections aide à mieux cerner leur cas d'utilisation et à estimer la pertinence de la structure. Les imbrications successives contribuent à une certaine forme de complexité mais n'implique pas forcement des requêtes moins performantes. Une collection très imbriquée peut être avantageuse si des requêtes prioritaires nécessitent une majorité des informations imbriquées. Si ce n'est pas le cas, l'impact des opérations de projection sera à prendre en compte (voir métriques suivantes) ainsi que la restructuration des données pour la réponse si le chemin d'accès de la requête et le sens d'imbrication des données ne coïncident pas.

Sur l'exemple, la profondeur des collections *Owners, Companies et Employees* est 0, 2, et 2 respectivement. La profondeur maximale du schéma est de 2. Notons que dans la collection *Employees*, le type *tCompany* n'ajoute pas de niveau d'imbrication, il ajoute uniquement un tableau avec des références sur *Owners*.

Profondeur d'un type de document : la métrique *docDepthInCol* (7) indique le niveau où apparaît un type de document t dans une collection φ. Si les éléments de la collection sont de type t (nœud $t@l_0$), la profondeur est zéro, sinon on cherche le niveau le plus profond où est imbriqué un document de ce type (nœuds $EMB\, t$) en suivant les chemins racine-feuilles.

$$docDepthInCol(\varphi, t) = \begin{cases} 0 & \text{le nœud fils de } \varphi \text{ est } t@l_0 \\ max(docDepth(p_i, t) & p_i \text{ est un chemin de la racine } \varphi \text{ à une feuille} \end{cases} \tag{7}$$

$$docDepth(p, t) = n \quad \text{nombre de nœuds } EMB \text{ entre la racine et } *\, EMB\, t \tag{8}$$

Par exemple, dans la collection *Employees*, le niveau d'imbrication de *tPublicity* est 2, celui de *tEmployee* est 0. *tEmployee* est aussi imbriqué au niveau 2 de la collection *Companies*. Nous introduisons également les métriques *maxDocDepth* (9) et *minDocDepth* (10) qui indiquent le niveau le plus et le moins profond où le type de document apparaît dans le schéma.

$$maxDocDepth(t) = max(DocDepthInCol(\varphi_i, t)) \quad \varphi_i \in x \ \wedge \ t \in \varphi_i \tag{9}$$

$$minDocDepth(t) = min(DocDepthInCol(\varphi_i), t) \quad \varphi_i \in x \ \wedge \ t \in \varphi_i \tag{10}$$

Connaître les niveaux minimum et maximum permet d'estimer combien de niveaux intermédiaires il faut traiter pour l'accès le plus ou le moins direct. Sur l'exemple, notons qu'il n'y a pas de collection de documents *tDepartment*, telle que *minDocDepth(tDepartment)* = 1.

4.3. *Largeur des documents*

Ici nous nous intéressons à la complexité d'un type de document en termes du nombre d'attributs et de leur type, atomique ou complexe (documents ou tableaux de documents imbriqués). Ces métriques sont motivées par le fait que des documents avec plusieurs attributs complexes peuvent induire des opérations d'accès et de projection plus conséquentes. En effet, pour extraire les attributs nécessaires à l'évaluation d'une requête, il est nécessaire d'enlever les autres attributs ce qui s'avère plus coûteux pour des

document "larges". Lors de l'évaluation d'un schéma, on pourra notamment analyser le choix d'une structure à la fois "large" et très imbriquée.

La métrique *docWidth* (11), reflète la "largeur" d'un type de document en associant un coefficient aux types d'attributs qu'il contient tels que le type atomique ($cfAtom$), le type document ($cfDoc$), le type tableau de valeurs atomiques ($cfTblAtom$) et tableau de documents ($cfTblDoc$). La "largeur" dépendra considérablement de ces coefficients et le nombre d'attributs correspondant à son type. Les quantités d'attributs par type sont fournies par les métriques *nbrAtomicAttributes*, *nbrDocAttributes*, *nbrArrayAtomicAttributes* et *nbrArrayDocAttributes* qui déterminent la quantité d'attributs atomiques, documents, tableaux de type atomique et tableaux de type document présents dans un type de document.

$$\begin{aligned} docWidth(t,\varphi) = {} & cfAtom * nbrAtomicAttributes(t,\varphi) + \\ & cfDoc * nbrDocAttributes(t,\varphi) + \\ & cfTblAtom * nbrArrayAtomicAttributes(t,\varphi) + \\ & cfTblDoc * nbrArrayDocAttributes(t,\varphi) \end{aligned} \tag{11}$$

$nbrAtomicAttributes(t,\varphi) = n$ nombre d'attributs de type atomique présents dans le type t (12)

$nbrDocAttributes(t,\varphi) = n$ nombre d'attributs de type document présents dans le type t (13)

$nbrArrayAtomicAttributes(t,\varphi) = n$ nombre d'attributs de type tableau de valeurs atomiques (14)

$nbrArrayDocAttributes(t,\varphi) = n$ nombre d'attributs de type tableau de documents (15)

Pour docWidth(), nous proposons d'attribuer les valeurs des coefficients en augmentant la valeur en fonction de la complexité du type des attributs afin de refléter leur complexité. Si $cfAtom$=1, $cfDoc$=2, $cfTblAtom$=1 et $cfTblDoc$=3, alors $cfTblDoc$ a le plus grand poids.

Les métriques indiquant le nombre d'attributs peuvent être utilisées séparément selon les analyses souhaitées. La taille des tableaux n'est pas prise en compte ici car elle n'est pas forcément disponible. Si c'est le cas, il semble intéressant de différencier les ordres de grandeur des tableaux. Des tableaux de taille 4 ou 5 sont du même ordre de grandeur, contrairement à des tableaux prévus pour de milliers d'éléments.

Les collections *Companies* et *Employees* de l'exemple, utilisent des documents de type *tDeparment* mais ils n'ont néanmoins pas les mêmes attributs. Dans *Employees* le type inclut des tableaux de compagnies et publicités, *docWidth(Employees,tDeparment)* = 8 , contrairement à *Companies* où *docWidth (Companies,tDeparment)* = 4.

4.4. *Taux de référencement*

Le maintien de l'intégrité référentielle devient un problème pour les collections dont les documents sont beaucoup référencés par de documents d'autres collections. Pour une collection avec des documents d'un certain type t, la métrique *refLoad* (16) indique le nombre d'attributs (d'autres types) qui sont des références potentielles sur les documents de type t.

$refLoad(\varphi) = n$ soit $t@l_0$ le nœud fils de φ, n est le nombre de nœuds $*REF\,t$ du schéma (16)

Pour la collection *Owners* de notre exemple, ses documents sont référencés par 2 collections : *Agencies* référence au niveau 0 en utilisant un tableaux de références alors que *Creatives* référence dans un document imbriqué au niveau 2.

4.5. *Métriques de redondance*

Nous nous intéressons ici à la redondance de données qui peut exister dans la base. La redondance des documents peut accélérer les accès et limiter certaines opérations coûteuses (par exemple des jointures). Cependant, elle impacte l'empreinte mémoire de la base et sa maintenabilité en matière de cohérence (complexité d'écriture des programmes et coût). Pour la métrique de redondance *docCopiesInCol* (17), nous utilisons l'information de cardinalité des associations conjointement avec les choix faits sur le schéma. La redondance apparaît pour certains cas de représentation de l'association par imbrication de documents.

$$docCopiesInCol(t,\varphi) = \begin{cases} 0 & : t \notin \varphi \; docExistence(\varphi,t)=0 \\ 1 & : \text{ le nœud fils de } \varphi \text{ est } t@l_0 \\ \prod card(r_{rol},t) & : \begin{matrix} r_{rol} \text{ parent de un nœud } EMB \\ \text{dans le chemin entre } \varphi \text{ et } *EMBt \end{matrix} \end{cases} \quad (17)$$

$$card(r,\varepsilon) = n \qquad \text{cardinalité de } r \text{ du côté } \varepsilon \text{ dans le modèle UML} \quad (18)$$

Dans la collection *Employées* du schéma de la Figure 2, l'attribut pour le département, nommé *dept*, introduit de la redondance pour les compagnies. Par l'association r1 une compagnie A peut être associée à n1 départements. Il y aura donc autant de copies du document A. Si de plus un département est référencé par n2 employées (association r2), alors il y aura n1 x n2 copies du document A.

Par ailleurs, nous proposons la métrique *docTypeCopies(t)* qui indique le nombre de fois qu'un type de document est utilisé dans le schéma. Ceci reflète le nombre de structures qui peuvent potentiellement stocker des documents de type t. Cette métrique utilise la métrique d'existence.

Synthèse

Les métriques structurelles introduites dans cette section mettent en évidence des caractéristiques qui peuvent avoir un impact notable sur plusieurs aspects liés aux données et aux programmes. Ces métriques proposées n'ont pas l'ambition de représenter un ensemble complet mais sont une base qui peut évoluer. Elles sont aussi un complément à d'autres mesures de performances opérationnelles si elles sont disponibles. Elles peuvent ainsi être utilisées pour analyser et comparer différentes manières de structurer les données en considérant des critères applicatifs.

5. Scénario de validation

La validation a été réalisée en suivant les trois phases de SCORUS mentionnées précédemment. Nous avons considéré plusieurs variantes de structuration de données, évalué les métriques structurelles et réalisé une analyse guidée pas les critère prioritaires d'un cadre applicatif. Il s'agissait en priorité de faire émerger le ou les schémas les plus favorables selon certains critères mais aussi d'écarter des choix très défavorables ou encore, d'envisager des schémas alternatifs qui n'étaient pas forcement considérés au

départ. Ceci est arrivé lors de notre expérimentation où l'une des alternatives générées automatiquement s'est avérée être pertinente alors qu'elle ne faisait pas partie des choix "naturels" du développeur.

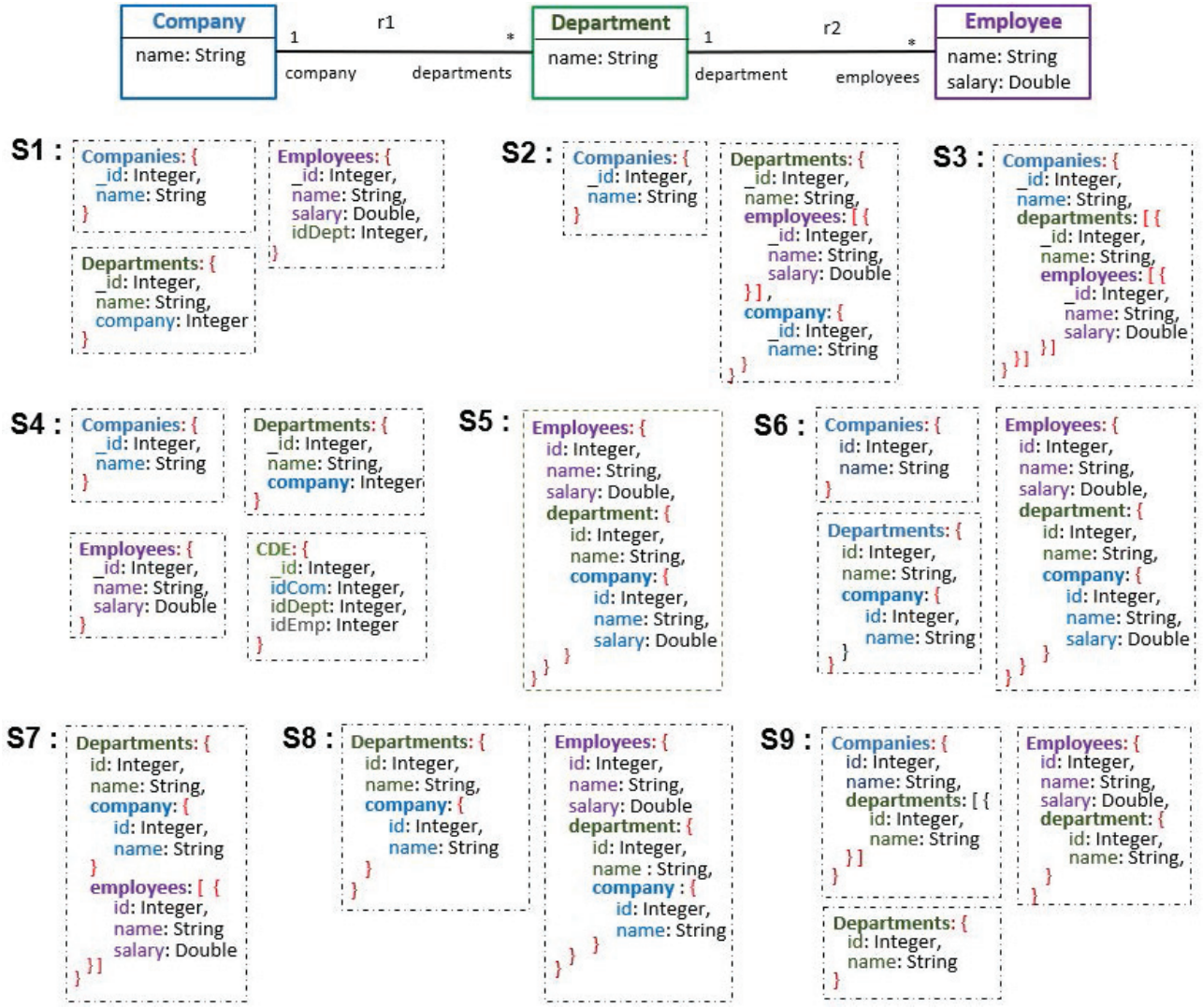


Figure 7. *Ensemble des structures étudiées*

Alternatives de structuration

Nous travaillons avec les AJSchémas $S = \{S1, \ldots, S9\}$, illustrés sur la Figure 7. Les AJSchémas $S1$ à $S6$ correspondent à une abstraction de la structuration des données des bases MongoDB utilisées dans (Gómez *et al.*, 2016) [5]. Les structurations proposées par les AJSchémas $S7$ à $S9$ n'avaient pas été envisagées précédemment.

Les AJSchémas de l'ensemble S présentent divers choix concernant l'existence de collections, l'imbrication de documents, le référencement et la duplication de documents. Dans certains cas il y a différents niveaux d'accès et plusieurs copies d'un même document. Les caractéristiques dominantes de ces AJSchémas sont :

— *Structuration avec référencement de documents et sans imbrication :* S1 et S4

— *Collection unique et imbrication totale des données :* S3, S5, S7

5. Version simplifiée du cas utilisé dans les sections précédentes.

— *Structuration avec imbrication et référencement de documents :* S2, S8, S9

— *Structuration avec imbrication de documents et duplication potentielle des documents imbriqués :* S5, S6, S7, S8

Métriques structurelles et critères applicatifs

D'un point de vue applicatif, nous disposons des informations suivantes afin d'établir les critères prioritaires permettant d'identifier des métriques utiles.

Les requêtes les plus prioritaires portent sur les entreprises et le nom de leurs départements (priorité forte) mais aussi pour connaître l'employé qui a le salaire le plus élevé dans l'entreprise en donnant l'identifiant de l'entreprise ou le nom de l'entreprise. Ces informations sur les accès prioritaires ainsi que d'autres informations permettent d'établir des critères d'analyse (cf. Tableau 2) . Tenant compte des accès prioritaires, les entreprises jouent un rôle important (critère1) ainsi que la facilité pour manipuler leurs instances (critère 5). Les départements sont accédés via les entreprises (critère 6). De plus, on sait que la cohérence des données sur les entreprises est importante. Il est donc préférable de limiter les copies des données des entreprises (critère 2). Par ailleurs, l'accès à l'ensemble des employés exclusivement n'est pas prioritaire (critère 4).

Critères à considérer		**Critères à considérer**
1	Favoriser l'existence de la collection **Companies**	*colExistence*
2	Éviter les copies de documents **tCompany**	*docCopies*
3	Réduire les références à la collection **Employees**	*refLoad*
4	L'existence d'une collection **Employees** n'est pas prioritaire	*colExistence*
5	Réduire la complexité de la collection **Companies**	*colExistence, docWidth*
6	Privilégier l'imbrication de **tDepartment** dans **Companies**	*colExistence, docExistence, docDepthInCol*

Tableau 2. *Critères applicatifs retenus et métriques utilisées*

Une fois les critères à prendre en compte définis, nous avons identifié une ou plusieurs métriques pertinentes à évaluer en fonction de chaque critère et en considérant des aspects structurels impliqués. Les critères 1, 4, 5 et 6 dépendent des informations des entreprises et des employés en tant que collections, la métrique *colExistence* est donc associée à ces critères. Comme nous nous intéressons à la cohérence des informations des entreprises à travers leurs copies potentielles, nous suggérons d'utiliser la métrique *docCopies* (critère 2). Afin d'identifier si les départements sont dans la collection Companies, nous proposons d'utiliser la métrique *docDepthInCol* qui permet de savoir si un type de document est imbriqué dans une collection (critère 6).

Évaluation des métriques structurelles

Nous avons évalué ces métriques pour les 9 variantes de structuration (cf. Figure 7). Les résultats de cette évaluation sont indiquées sur le Tableau 3. Cette tableau montre, par exemple, l'évaluation de la métrique *docCopies* sur le type *tCompany* pour l'ensemble des AJSchémas. Cette métrique indique le nombre de collections où le type de document *tCompany* est présent. Notons que le type *tCompany* est présent dans une seule collection dans chaque schéma, à l'exception du schéma $S6$, où il existe dans trois collections. Les cases grises indiquent que la métrique ne s'applique pas à ce AJSchéma.

Métriques \ Schéma	S1	S2	S3	S4	S5	S6	S7	S8	S9
colExistence(tCompany)	1	1	1	1	0	1	0	0	1
docCopies(tCompany)	1	1	1	1	1	3	1	1	1
refLoad(Employees)	0			1	0	0		0	0
colExistence(tCompany)	1	0	0	1	1	1	0	0	1
docWidth(Companies,l1)	1	1	3	1		1			3
docExistence(tDepartment,Companies)	0	0	1	0		0			1

Tableau 3. *Évaluation des schémas*

Analyse d'alternatives de structuration

La formalisation des critères et leur évaluation pour chaque AJSchéma sont illustrées dans le Tableau 4. Chaque critère est formalisé comme une fonction dont la valeur est à maximiser ou à minimiser. Les valeurs des critères ont été normalisées entre 0 et 1. Par exemple, le critère 2 indique qu'il est important de limiter le nombre de copies des données des entreprises. Ceci est traité en minimisant la valeur de $docCopiestCompany(s)$. L'AJSchéma qui aura la plus petite valeur pour cette métrique est considéré comme le plus intéressant et gagnera un maximum de points pour le critère 2. Ici, pour le critère 2, tous les AJSchémas ont la valeur maximale 1 excepté S6. En effet, S6 stocke des entreprises dans trois collections ce qui est interprété comme trois copies potentielles.

	Critère \ Schéma	S1	S2	S3	S4	S5	S6	S7	S8	S9
1	$f_{c_1}(s) = colExistenceCompanies(s)$	1.00	1.00	1.00	1.00	0.00	1.00	0.00	0.00	1.00
2	$f_{c_2}(s) = docCopiestCompany^{min}(s)$	1.00	1.00	1.00	1.00	1.00	0.33	1.00	1.00	1.00
3	$f_{c_3}(s) = refLoadEmployees^{max}(s)$	1.00	1.00	1.00	0.00	1.00	1.00	1.00	1.00	1.00
4	$f_{c_4}(s) = colExistenceEmployees(s)$	1.00	0.00	0.00	1.00	1.00	1.00	0.00	0.00	1.00
5	$f_{c_5}(s) = levelWidthCompaniesL_1{}^{min}(s)$	1.00	1.00	0.33	1.00	0.00	1.00	0.00	0.00	0.33
6	$f_{c_6}(s) = docDptInCompanies^{min}(s)$	0.00	0.00	1.00	0.00	0.00	0.00	0.00	0.00	1.00

Tableau 4. *Critères*

L'évaluation de chaque critère introduit un ordre relatif entre les AJSchémas et permet de les classer en fonction de chacun d'eux. Par exemple, au regard du critère 4, les AJSchémas S1, S4, S5, S6, S8 et S9 sont à privilégier par rapport aux autres.

Cependant, le fait de privilégier complètement un AJSchéma dépend de l'ensemble des critères. Par conséquent, pour prendre des décisions en les impliquant tous, nous adoptons une analyse multi-critères basée sur une somme pondérée des critères par AJSchéma.

L'importance de chaque critère est reflétée par son poids dans la somme pondérée. La fonction d'évaluation d'un AJSchéma, noté *schemaEvaluation* (19) fait la somme pondérée des critères. Une évaluation élevée reflète une bonne adaptation d'un AJSchéma aux besoins considérés.

$$schemaEvaluation(s) = \sum_{i=1}^{|Criteria|} factor_{criterion_i} * f_{criterion_i}(s) \quad (19)$$

Nous avons considéré trois cas qui ont donné lieu à trois pondérations différentes introduites dans le Tableau 5 :

— Cas 1, tous les critères ont la même importance.

— Cas 2, priorité aux critères concernant la facilité d'utilisation de l'extension des entreprises.

— Cas 3, ajout d'une priorité de la collection Employees en supposant qu'il est motivé par un nouveau patron d'accès important.

Critère	**Facteur**		
	Cas 1	Cas 2	Cas 3
Critère 1	16.667	50	30
Critère 2	16.667	10	10
Critère 3	16.667	0	0
Critère 4	16.667	0	20
Critère 5	16.667	15	15
Critère 6	16.667	25	25

Tableau 5. *Facteur d'évaluation des 6 critères pour les cas retenus*

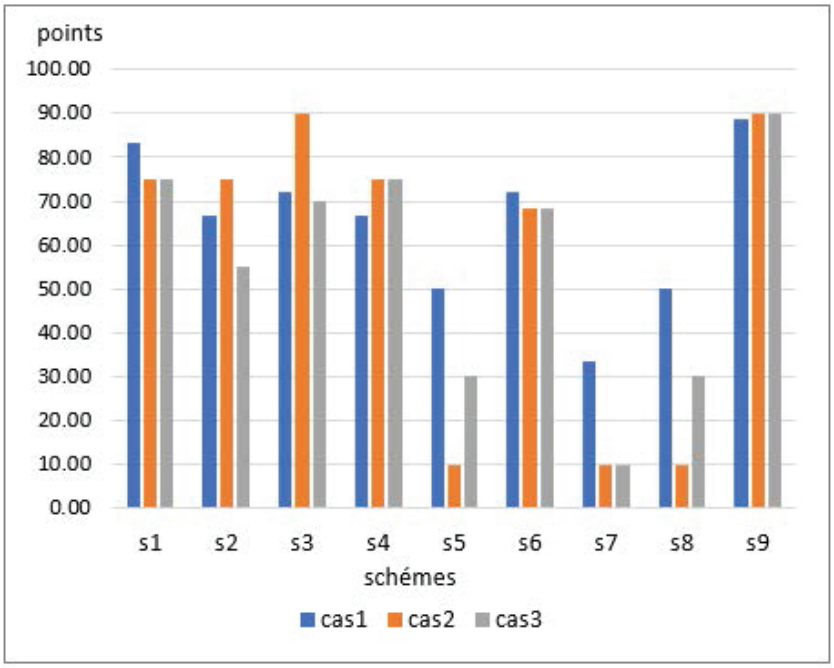
Le cas 2 représente le fait que les requêtes les plus prioritaires portent sur les entreprises et les noms de leurs départements. Ceci est exprimé par le critère C1 mais aussi, en partie, avec les critères C5 et C6. Ainsi, les pondérations proposées dans ce cas donnent beaucoup de poids à C1 et renforcent C5 et C6 par rapport aux autres critères.

Dans le cas où l'on ne disposerait pas d'information prioritaire sur un critère, on utiliserait une pondération uniforme pour tous les critères comme dans la cas 1. Ceci peut aussi être motivé par le fait que les critères considérés représentent des priorités qui alternent dans le temps. Les critères ne sont pas tous prioritaires au même moment. Dans ce cas, on peut préférer une structuration qui représente un compromis.

Les cas 3 est introduit dans cet étude pour considérer une situation où, en plus des priorités déjà connues, on doit également faciliter l'accès aux données des Employées. Dans ce cas, le critère 4 devient important contrairement à la situation du cas 2 qui est "dominé" par le critère 1. Ce critère 1 est indispensable pour l'accès direct à l'information des entreprises.

La figure 8 montre le résultat de l'évaluation des 9 AJSchémas pour les trois cas. Les évaluations placent les AJSchémas S5, S7, S8 comme les moins bons dans les trois cas considérés. La structuration dans S5 et S7 se base sur une seule collection qui n'est pas prioritaire dans les critères pris en compte. Alors que S3 se démarque dans le cas 2, pour la forte priorité de la collection *Companies*, seule collection de S3 qui imbrique des documents en accord avec les critères recherchés.

S9 et S6, parmi d'autres schémas, sont stables dans leurs scores pour les trois cas. S9 est le meilleur car il correspond à tous les critères. Les bons résultats aux trois cas dénotent une forme de "polyvalence" du schéma qui permet de résister aux évolutions des priorités. S6 paie le coût de ne pas considérer l'imbrication dans la collection *Agencies* et d'introduire de la redondance de documents alors qu'on préfère l'éviter (critère 2).



Schema / Cas	s1	s2	s3	s4	s5	s6	s7	s8	s9
Cas 1	83.33	66.67	72.22	66.67	50.00	72.22	33.33	50.00	88.89
Cas 2	75.00	75.00	90.00	75.00	10.00	68.33	10.00	10.00	90.00
Cas 3	75	55	70	75	30	68.3	10	30	90

Figure 8. *Évaluation de 9 AJSchémas selon les trois cas retenus*

Les critères à considérer et leur poids [6] dépendent du contexte applicatif mais peuvent aussi correspondre à de bonnes pratiques préconisées pour le développement ou à une priorité générale. Par exemple, adopter des structures très "compactes" pour limiter l'empreinte mémoire lorsque des données seront peu utilisées. Ou, en donnant priorité à la qualité logicielle, privilégier les schémas les plus "lisibles". Sachant que les critères peuvent diverger et évoluent certainement, l'utilisation des métriques et critères pour un choix de schéma peut aider dans un processus continu de "tuning" de la base qui peut conduire à des évolutions de la structuration ou à la création de copies des données avec des structures différentes. Pendant un certain temps, une base pourrait avoir, pour les mêmes données, une copie avec une schéma Sx et une autre copie avec un schéma Sy.

6. Travaux connexes

Dans l'étude de l'état de l'art, nous nous sommes intéressés aux travaux concernant des systèmes NoSQL ainsi qu'à des propositions antérieures pour des données complexes, des documents XML et des métriques logicielles.

Concernant les travaux sur des données semi-structurées, (Klettke *et al.*, 2002) s'appuie sur le modèle de qualité de software ISO 9126 et adapte cinq métriques pour évaluer des documents XML en travaillant sur la DTD. Ces travaux sont basés sur une représentation sous forme de graphe et les métriques considèrent le nombre de références, nœuds et font un rapprochement avec la complexité cyclomatique

6. Le choix des pondérations reste un sujet à approfondir

(McCabe, 1976). (Pušnik *et al*., 2014) propose 6 métriques associées, chacune à un aspect de qualité telles que la structure, la clarté, l'optimalité, le minimalisme, la réutilisation et la flexibilité. Ces métriques utilisent 25 variables qui mesurent le nombre d'éléments, d'annotations, de références et de types parmi d'autres. Ces travaux nous ont servi de point de départ, nous avons étendu et adapté ces propositions pour prendre en compte les particularités de JSON en tant que documents incorporés et types d'attributs complexes.

Nos métriques sont également influencées par les métriques logicielles (Li, Henry, 1993 ; Chidamber, Kemerer, 1991 ; McCabe, 1976). Leur métriques reflètent, entre autres, les niveaux de couplage entre les composants, la taille des hiérarchies, la taille des objets et le nombre de méthodes. (Timóteo *et al*., 2008) analyse les métriques logicielles basées sur la complexité des concepts de code et orientées objet. Nous avons été particulièrement intéressés par les métriques statiques. Nous avons notamment pris en compte les métriques de cohésion et de couplage, largement adoptées en génie logiciel car elles nous ont semblé intéressantes pour les approches orientées document.

Certains travaux s'intéressent à la modélisation de données pour les bases NoSQL. La motivation de la plupart de ces travaux est le choix, presque ad hoc, d'une structure de données favorable aux performances d'exécution d'un ensemble de requêtes donné. Zhao et al. (Zhao *et al*., 2014) proposent un algorithme pour la création systématique d'une base de données orientées documents à partir du modèle entité-relation. Cet algorithme propose une dé-normalisation de ce modèle avec pré-calcul des jointures naturelles par imbrication de documents. Le schéma résultant correspond au modèle du schéma S6 de notre scénario de validation. Ce choix engendre en général de la redondance de données. Les auteurs proposent une métrique pour cela en utilisant la connaissance du volume des données. Nous proposons un ensemble plus large de métriques structurelles qui inclut deux métriques pour analyser la redondance sans connaissance du volume des données. Des travaux tels que (Mior *et al*., 2017) et (Lombardo *et al*., 2012) s'intéressent au modèle orienté colonnes dans Cassandra. Les objectifs des études portent sur le stockage et les performances des requêtes. Ils créent un schéma en considèrant un ensemble de requêtes. Lombardo et al.(Lombardo *et al*., 2012) proposent des structures différentes, chacune adaptée à l'évaluation d'une requête. Mior et al. (Mior *et al*., 2017) proposent des structures adaptées pour l'évaluation de plusieurs requêtes. Comme nous, ces travaux cherchent à trouver la structuration de données la mieux adaptée aux besoins de l'application. Par contre, nous créons plusieurs alternatives de structuration afin de les évaluer à l'aide de nos métriques structurelles, avant de prendre une décision finale. Les travaux présentés dans (Abdelhedi *et al*., 2017) et (Atzeni *et al*., 2016) rassemblent les concepts de différentes familles de données NoSQL afin de créer à partir d'un modèle UML, des alternatives de structuration selon le système cible. Abdelhedi et al. (Abdelhedi *et al*., 2017) créent comme nous, plusieurs alternatives de structuration pour des systèmes orientés documents, mais aussi pour des autres systèmes tels que orientés colonnes et graphes. La validation de ces travaux repose essentiellement sur le temps de réponse des requêtes, nous proposons un ensemble de métriques qui permet d'évaluer les structures statiquement.

Des travaux récents s'intéressent à l'analyse du schéma d'une base de données déjà implémentée. Les travaux de Ruiz et al. (Ruiz *et al*., 2015), Wang (Wang *et al*., 2015) et Gallinucci (Gallinucci *et al*., 2018) portent sur la rétro-ingénierie afin d'obtenir des informations sur la structuration des données. Ils portent essentiellement sur des documents style JSON. Ces travaux utilisent des terminologies différentes mais les concepts sont proches. Ils extraient les attributs utilisés dans les documents et tentent d'identifier leur type. En tenant compte du caractère semi-structuré et de l'hétérogénéité des documents, chaque proposition extrait des informations qui reflètent les variantes présentes dans la base. Parmi les outils existants

travaillant sur des bases opérationnelles, notons MongoDBCompass (*MongoDBCompass*, s. d.) qui permet de monitorer le temps d'exécution des requêtes, de connaître le volume des données d'une collection de documents et d'extraire des informations par rapport à la structure d'une collection. Certains outils se sont inspirés de l'approche JSON schema (jsonSchema, s. d.) pour faciliter l'analyse des documents JSON et pouvoir abstraire un "schéma" avec des définitions explicites de collection et de type. Bien que ces approches de rétro-ingénierie partent de la direction opposée à la nôtre (à partir d'une base mise en œuvre), elles visent comme nous la motivation d'aider à comprendre la structuration des données et à utiliser des critères pour évaluer les structures reconstruites. Nous avons tenu compte de leurs critères pour valider et comparer nos métriques notamment celles qui concernent l'existence de collections et types de documents.

D'autres travaux, sans formaliser ou suggérer des métriques sur des schémas semi-structurés, fournissent des lignes directrices, des bonnes pratiques et des aspects à prendre en compte dans le choix des structures. Sadalage et al. (Sadalage, Fowler, 2012) s'intéressent aux problèmes liés à la migration d'une base de données relationnelle vers des colonnes, documents et graphes. (Copeland, 2013 ; *RDBMS to MongoDB Migration Guide*, 2017) propose des directives pour la création de bases de données Mongo basées sur plusieurs cas d'utilisation. Abiteboul (Abiteboul, 1997) fournit des aspects à considérer pour les données semi-structurées et un aperçu de propositions de modèles et de langages Nous avons étudié ces travaux afin de les formaliser dans les critères que nous proposons et de les prendre en compte dans l'analyse des schémas.

7. Conclusion et perspectives

Dans ce travail, nous nous sommes intéressés à des questions de qualité des structures de données pour des bases de documents JSON, telles que MongoDB. La flexibilité de structuration de ces bases est appréciée par la souplesse qu'elle permet pour représenter des données semi-structurées. Cependant, cette flexibilité a un coût dans les performances, le stockage, la lisibilité et la maintenabilité des bases et des applications. Ainsi, le choix de la structuration des données est très important et ne doit pas être négligé.

Dans cet article, nous avons présenté une vision globale du projet SCORUS, qui vise à aider l'utilisateur à clarifier les possibilités de structuration des données orientées document et à fournir des métriques pour prendre des décisions de manière plus consciente. Nous avons décrit brièvement la génération automatique d'alternatives de structuration et nous avons défini des métriques structurelles pour des "schémas" JSON. Ces métriques, organisées en catégories, reflètent des éléments de complexité du schéma qui jouent sur des aspects de qualité de la base. Elles couvrent des aspects tels que l'existence, la profondeur d'imbrication, la largeur d'imbrication, la référence et la redondance. Les métriques fournissent des éléments quantitatifs qui peuvent être utilisés pour analyser et comparer différentes manières de structurer les données.

Nous avons présenté un scénario d'utilisation des métriques avec plusieurs variantes de schémas et certains critères et priorités applicatifs. L'analyse avec les critères, permet d'écarter certains schémas et d'en mettre en avant d'autres. Ces résultats sur les aspects structurels ont été comparés et, sont bien en phase, avec les résultats d'expériences d'évaluation de performances que nous avons menées avec des bases contenant des données. Il est intéressant de noter que lors du travail sur les structures nous avons

pu considérer à "faible coût" plus de variantes de schémas que lors de l'expérimentation avec les bases. Cela a apporté un résultat inattendu qui est l'identification d'un schéma différent avec de très bonnes caractéristiques.

Les propositions de génération d'alternatives de structuration et l'évaluation automatique des métriques sont opérationnelles dans le prototype *ScorusTool*. Pour un développeur, il suffit de donner un modèle UML et en un clic il peut visualiser l'ensemble de variantes de structures et leur métriques. En très peu de temps, il est possible de considérer de nombreuses alternatives qui ensuite sont analysées plus finement.

Les métriques proposées n'ont pas l'ambition de représenter un ensemble complet mais sont une base qui peut évoluer. La suite des travaux, inclut des validations à plus grande échelle. Il serait intéressant de réaliser une expérimentation avec *ScorusTool* en le proposant à des développeurs d'applications utilisant MongoDB.

A moyen terme nous proposons de travailler sur une approche de rétro-ingénierie de bases de documents, consistant à déduire l'AJSchéma correspondant à une base MongoDB existante. Cela permettrait d'évaluer les métriques sur ces AJSchémas et contribuer à une meilleure compréhension des bases et des applications. Une autre perspective, à plus long terme, est la formalisation d'un système de recommandations pour faciliter la définition des critères en utilisation les métriques, les requêtes fréquentes et autres préférences fonctionnelles ou non fonctionnelles des utilisateurs potentiels.

Remerciements

Nous remercions G. Vega, J. Chavarriaga, C. Labbé et J. Giraudin pour les échanges autour de ce travail ainsi qu'aux relecteurs anonymes pour leur retours.

Bibliographie

Abdelhedi F., Brahim A. A., Atigui F., Zurfluh G. (2017). Mda-based approach for nosql databases modelling. In *International conference on big data analytics and knowledge discovery*, p. 88–102.

Abiteboul S. (1997). Querying semi-structured data. In *Proceedings of the 6th international conference on database theory*, p. 1–18. London, UK, Springer-Verlag. Consulté sur http://dl.acm.org/citation.cfm?id=645502.656103

Atzeni P., Bugiotti F., Cabibbo L., Torlone R. (2016). Data modeling in the nosql world. *Journal : Computer Standards & Interfaces*.

Chidamber S. R., Kemerer C. F. (1991). Towards a metrics suite for object oriented design. *SIGPLAN Not.*, vol. 26, n° 11, p. 197–211. Consulté sur http://doi.acm.org/10.1145/118014.117970

Copeland R. (2013). *Book : Mongodb applied design patterns*. Oreilly.

Gallinucci E., Golfarelli M., Rizzi S. (2018). Schema profiling of document-oriented databases. *Journal : Information Systems*, vol. 75, p. 13–25.

Gómez P., Roncancio C., Casallas R. (2018). Métriques structurelles pour l'analyse de bases orientées documents. In *Actes du xxxvième congrès INFORSID.*

Gómez P., Roncancio C., Casallas R. (2019). Génération automatique d'alternatives de structuration de données pour systèmes orientés document. In *Actes du xxxviième congrès INFORSID.*

Gomaa H. (2005). *Book : Designing software product lines with uml*. IEEE.

Gomez P. (2018). *Analyse et évaluation de structures orientées document*. Thèse. Université Grenoble Alpes. https ://www.theses.fr/2018GREAM076.

Gómez P., Casallas R., Roncancio C. (2016). Data schema does matter, even in nosql systems ! In *Research challenges in information science (RCIS), tenth international conference on*, p. 1–6. Grenoble, France, IEEE.

JSON schema. (s. d.). *Json schema.* http ://json-schema.org/. (Accessed : 2019-05-07)

Kang K. C., Lee J., Donohoe P. (2002). Feature-oriented product line engineering. *Journal : IEEE software*, vol. 19, n° 4, p. 58–65.

Klettke M., Schneider L., Heuer A. (2002). Metrics for xml document collections. In *International conference on extending database technology*, p. 15–28.

Li W., Henry S. (1993). Object-oriented metrics that predict maintainability. *Journal of systems and software*, vol. 23, n° 2, p. 111–122.

Lombardo S., Nitto E. D., Ardagna D. (2012). Issues in handling complex data structures with nosql databases. In *14th international symposium on symbolic and numeric algorithms for scientific computing, SYNASC, timisoara, romania, september 26-29, 2012*, p. 443–448. Consulté sur http://dx.doi.org/10.1109/SYNASC.2012.59

McCabe T. J. (1976). A complexity measure. *Journal : IEEE Transactions on software Engineering*, n° 4, p. 308–320.

Mior M. J., Salem K., Aboulnaga A., Liu R. (2017). Nose : Schema design for nosql applications. *Journal : IEEE Transactions on Knowledge and Data Engineering*, vol. 29, n° 10, p. 2275–2289.

Mongodbcompass. (s. d.). https ://docs.mongodb.com/compass/master/. (Accessed : 2019-05-07)

Pušnik M., Heričko M., Budimac Z., Šumak B. (2014). Xml schema metrics for quality evaluation. *Journal : Computer science and information systems*, vol. 11, n° 4, p. 1271–1289.

Rdbms to mongodb migration guide. (2017, Nov). White Paper. Consulté sur https://www.mongodb.com/collateral/rdbms-mongodb-migration-guide

Ruiz D. S., Morales S. F., Molina J. G. (2015). Inferring versioned schemas from nosql databases and its applications. In *Conceptual modeling (ER)*, p. 467–480. Springer.

Sadalage P. J., Fowler M. (2012). *Nosql distilled : a brief guide to the emerging world of polyglot persistence, book*. Pearson Education.

Timóteo A. L., Álvaro A., De Almeida E. S., Lemos Meira S. R. de. (2008). *Software metrics : A survey.* Citeseer.

Wang L., Zhang S., Shi J., Jiao L., Hassanzadeh O., Zou J. *et al.* (2015). Schema management for document stores. *Proceedings of the VLDB Endowment*, vol. 8, n° 9, p. 922–933.

Zhao G., Lin Q., Li L., Li Z. (2014). Schema conversion model of sql database to nosql. In *P2p, parallel, grid, cloud and internet computing (3pgcic), ninth international conference on*, p. 355-362.

