Raisonnement Numérique sur les Relations d'Ordre pour le Web Sémantique

Chimène Fankam Stéphane Jean Guy Pierra
Laboratoire d'Informatique Scientifique et Industrielle
LISI/ENSMA and University of Poitiers
BP 40109, 86961 Futuroscope Cedex, France
{fankam,jean,pierra}@ensma.fr

ABSTRACT

Le Web Sémantique vise à permettre l'intégration et le partage d'informations entre différentes applications et organisations en utilisant des annotations basées sur des instances d'ontologies. Avec l'augmentation du volume de ces données, deux problèmes doivent être traités : (1) le problème du passage à l'échelle (*scalabilité*) et (2) le problème des performances des raisonnements. Les raisonneurs en mémoire centrale permettent de réaliser des raisonnements très efficacement, mais ils supportent difficilement des données de grande taille. Cet article discute de l'utilisation de bases de données, plus précisément de *Bases de Données à Base Ontologique* (BDBO), pour gérer les données d'annotation du Web Sémantique. De tels systèmes sont en mesure de gérer des données de taille réelle. Par contre, leur principale faiblesse est leur faible capacité de raisonnement et de déduction. C'est pourquoi nous proposons une approche qui consiste à enrichir les instances d'annotation avec des propriétés numériques ou alphanumériques permettant de remplacer certains raisonnements déductifs par des requêtes numériques. Les raisonnements auxquels nous nous intéressons ici sont les relations d'ordre qui recouvrent en particulier la subsomption, les taxonomies d'inclusion ensembliste et l'inclusion spatiale. Nous définissons formellement comment cette approche peut être implémentée et nous proposons également une extension des langages d'ontologies pour permettre de représenter explicitement ces cas. Les ontologies étant stockées dans les BDBO, ces extensions permettent au système de gestion des BDBO de réaliser dynamiquement l'enrichissement des instances et de réécrire les requêtes en des requêtes numériques. Nous présentons son implémentation sur OntoDB, une BDBO développée dans notre laboratoire.

1. INTRODUCTION

Le Web Sémantique est issu d'un effort du W3C pour permettre l'intégration de sources de données sur le Web. Pour rendre la sémantique des informations accessible à une machine, les ressources Web sont annotées avec des termes décrits comme des instances d'ontologies. Ces instances d'ontologies sont appelées *données à base ontologique*. Les technologies du Web Sémantique devenant matures et standardisées, elles sont utilisées dans des applications industrielles. En conséquence, une quantité croissante de données à base ontologique devient disponible sur le Web. Gérer de telles données pose deux problèmes majeurs :

- *le problème du passage à l'échelle (scalabilité).* Beaucoup d'applications ont besoin de gérer une quantité de données qui ne tient plus en mémoire centrale ;
- *le problème des performances des raisonnements.* Une ontologie est une conceptualisation basée sur une sémantique formelle permettant de raisonner sur les concepts et individus d'une ontologie. Ces opérations de raisonnement doivent pouvoir être réalisées en un temps raisonnable.

Résoudre ces deux problèmes est une condition essentielle pour réaliser la vision du Web Sémantique. La difficulté est de les résoudre conjointement. En effet, durant ces dernières années, plusieurs travaux se sont intéressés au problème de scalabilité en utilisant des bases de données. Si certains travaux se sont focalisés sur l'utilisation de bases de données pour stocker les données de niveau instance [15, 1], d'autres ont proposé de nouvelles architectures de bases de données pour stocker à la fois les données de niveau ontologie et de niveau instance. Nous appelons ces architectures de bases de données des *Bases de Données à Base Ontologique* (BDBO) [3, 5, 13, 16, 9]. L'évaluation de leurs performances a montré que certaines architectures passent relativement bien à l'échelle et permettent de gérer des données du Web Sémantique de taille réelle [19, 16, 9]. Cependant, si le passage à l'échelle constitue un atout majeur des bases de données, ce n'est pas le cas de leurs capacités de déduction. Ainsi, plusieurs propositions ont été faites pour combiner des raisonneurs avec des bases de données [14, 20, 4, 15]. Pour améliorer le temps de réponse de ce type d'architecture, le raisonnement est souvent effectué hors-ligne [16]. Ensuite, les bases de données sont utilisées pour matérialiser l'ensemble des faits déduits, ce qui conduit à un fort coût de stockage supplémentaire.

Dans cet article, nous proposons une approche alternative. L'idée est d'utiliser la capacité des bases de données de pouvoir traiter efficacement les requêtes numériques ou alphanumériques afin de réaliser des opérations de raison-

JFO 2008 December 1-3, 2008, Lyon, France

nements lors de l'exécution d'une requête. Ainsi, notre approche consiste à (1) utiliser la représentation des ontologies disponibles dans une BDBO pour interpréter sémantiquement le langage de manipulation et d'interrogation de la base de données et (2) enrichir les instances d'annotation avec de nouvelles valeurs de propriétés pour remplacer le raisonnement déductif par le traitement de requêtes numériques (ou alphanumériques). Par exemple, lorsqu'une propriété de type objet (`ObjectProperty`) π est définie en utilisant les caractéristiques de OWL2 `asymmetric`, `transitive` et `inverse functional`, définissant ainsi un ordre que nous qualifierons d'arborescent, cet ordre arborescent ($\prec$) peut être représenté par un intervalle numérique [2]. Ainsi, (1) lorsqu'une instance présentant une valeur pour la propriété π est insérée dans la base de données, deux valeurs de propriétés additionnelles (`lo_bound`, `hi_bound`) sont calculées par le système. Ces valeurs de propriétés reflètent l'ordre arborescent, c'est-à-dire que :

$$\mathtt{x} \prec \mathtt{y} \Leftrightarrow \mathtt{lo\_bound(y)} < \mathtt{lo\_bound(x)} < \mathtt{hi\_bound(x)} < \mathtt{hi\_bound(y)}.$$

Ensuite, (2) lorsque des instances d'annotation plus petites qu'une instance donnée sont recherchées, l'interpréteur de requêtes accède à l'ontologie et réécrit la requête récursive utilisant π en une requête numérique utilisant `lo_bound` et `hi_bound`. Ce type d'index n'est pas nouveau. En effet, plusieurs approches [2, 7], connues sous le nom *d'étiquetage* (*"labeling"*), ont proposé de calculer la fermeture transitive de relations en les indexant par des labels numériques ou alphanumériques. Cependant, ces approches sont souvent codées en dur dans le système de gestion de données pour des relations prédéfinies telles que le polymorphisme au travers de la subsomption de classes. Nous proposons des extensions aux langages d'ontologies qui permettent (1) d'identifier les situations où ce type d'approche peut être suivi et (2) de l'implémenter dynamiquement lorsqu'une ontologie est chargée. La formalisation proposée intègre diverses techniques d'étiquetage permettant de raisonner sur les différents types de relations récursives d'inclusion. Ceci recouvre le raisonnement sur les sous-classes et les requêtes taxonomiques, très utilisé dans l'annotation de ressources. De plus, nous montrons que cette formalisation peut également être utilisée efficacement pour les structures de DAG (Graphe Dirigé Acyclique) employées dans les applications spatiales et temporelles.

Cet article est organisé comme suit. Dans la section suivante, nous présentons une vue d'ensemble des BDBO. Nous proposons une taxonomie des BDBO existantes et présentons leurs capacités de passage à l'échelle et de déduction. Dans la section 3, nous présentons une formalisation pour transformer un raisonnement déductif en un raisonnement numérique pour les propriétés définissant un ordre partiel sur des ensembles. L'implémentation de cette formalisation au sein des BDBO est discutée à la section 4 et l'implémentation que nous avons réalisée sur une application réelle est décrite dans la section 5. Enfin, nous concluons dans la section 6 et discutons des perspectives de ce travail.

2. LES BASES DE DONNÉES À BASE ONTOLOGIQUE (BDBO)

Au cours de ces dernières années, plusieurs architectures de BDBO ont été proposées. Nous présentons d'abord une proposition de taxonomie de ces architectures. Puis, nous discutons de leurs capacités pour résoudre les problèmes de passage à l'échelle et de raisonnement qui se posent dans le contexte du Web Sémantique.

2.1 Une Classification des BDBO

Les BDBO permettent de stocker différentes catégories de données, en particulier, les données de niveau ontologie et les données de niveau instance. Ces données peuvent être stockées en utilisant un nombre variable de schémas. C'est pourquoi nous proposons de classifier les architectures de BDBO en fonction du nombre de schémas utilisés.

BDBO de type 1. Dans les BDBO de type 1, les informations sont représentées en utilisant un seul schéma composé d'une unique table de triplets `(subject, predicate, object)` [10, 21, 6, 17]. Cette table, qualifiée de *table verticale* [2], peut être utilisée à la fois pour les données de niveau ontologie et de niveau instance. Pour les données de niveau ontologie, les trois colonnes de cette table représentent respectivement l'identifiant d'un élément d'ontologie, un prédicat et la valeur du prédicat qui peut être soit un identifiant d'un élément d'ontologie, soit une valeur littérale. Par exemple, le triplet [1] `(Student, subClassOf, Person)` représente une relation de subsomption entre les classes `Student` et `Person`. Pour les données de niveau instance, les trois colonnes de cette table représentent respectivement l'identifiant d'une instance, une caractéristique d'une instance (c'est-à-dire, une propriété où l'appartenance à une classe) et la valeur de cette caractéristique. Par exemple, le triplet `(Peter, grade, PhD)` représente le fait que Peter a le grade de docteur. Pour illustrer les différentes approches de stockage proposées par les BDBO, nous utilisons l'ontologie très simple présentée sur la figure 1. Cette ontologie (partie du haut) est représentée avec des instances (partie du bas) sous la forme d'un graphe. La figure 2 présente un exemple simpliste d'une ontologie (partie du haut) . Un extrait de la table verticale correspondante est présenté sur la figure 2.

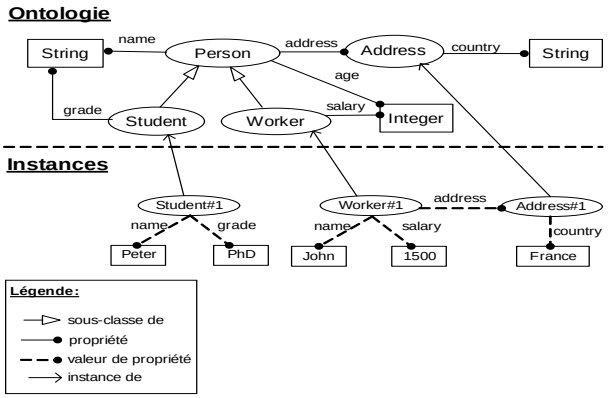

Figure 1: Exemple simpliste d'une ontologie

BDBO de type 2. Les BDBO de type 2 stockent séparément les données de niveau ontologie et de niveau instance dans deux schémas distincts [3, 5, 13]. Le schéma pour les données de niveau ontologie dépend du modèle d'ontologies

1. RDF utilise des URI comme identifiants. Pour des raisons de lisibilité nous utilisons des noms dans cet article.

TRIPLES			
Subject	**Predicate**	**Object**	
Person	rdf:type	rdfs:Class	Ontologie
Student	rdf:type	rdfs:Class	
Student	rdfs:subClassOf	Person	
name	rdf:type	rdf:Property	
name	rdfs:range	xsd:String	
…	…	…	
Student#1	rdf:type	Student	Instances
Student#1	name	Peter	
Student#1	grade	PhD	
Worker#1	address	Address#1	
…	…	…	

Figure 2: Approche des BDBO de type 1

utilisé pour représenter les ontologies (par exemple, RDFS, OWL ou PLIB). Il est composé de tables utilisées pour stocker chaque constructeur d'ontologies tel que les classes, les propriétés ou les relations de subsomption. Pour les données de niveau instance, plusieurs schémas ont été proposés. Une table verticale peut être utilisée pour stocker les données de niveau instance comme des triplets [13, 5]. Une alternative consiste à utiliser une *représentation binaire* où chaque classe est représentée par une table unaire et chaque propriété par une table binaire [3, 5, 15, 1]. Récemment, une approche appelée *table par classe* a été proposée. Elle consiste à associer à chaque classe une table d'instance ayant une colonne pour chaque propriété associée à une valeur pour au moins une instance de la classe [9, 16]. Ces trois principales approches ont également des variantes (voir [19] pour plus de détails).

La figure 3 présente un exemple de BDBO de type 2 stockant les données de l'exemple précédent (voir figure 2). Dans cet exemple, les données de niveau ontologie sont stockées en utilisant un schéma pour des ontologies RDFS. Dans la partie basse de cette figure, les données de niveau instance sont représentées en utilisant une représentation binaire.

Ontologie

Class		SubClassOf		Property		Domain		Range	
ID	**Name**	**Sub**	**Sup**	**ID**	**Name**	**prop**	**class**	**prop**	**type**
1	Person	2	1	1	name	1	1	1	xsd:string
2	Student	3	1	2	age	2	1	2	xsd:integer
3	Worker			3	grade	3	2	3	xsd:string
4	Address			…	…	…	…	…	…

Instances

Person	
ID	

Student	
ID	
Student#1	

Name	
ID	**Value**
Student#1	Peter
Worker#1	John

Grade	
ID	**Value**
Student#1	PhD

Address	
ID	
Address#1	

Worker	
ID	
Worker#1	

Salary	
ID	**Value**
Worker#1	1500

Country	
ID	**Value**
Address#1	France

Figure 3: Approche des BDBO de type 2

BDBO de type 3. OntoDB [9, 18] propose l'ajout d'un autre schéma aux BDBO de type 2. Ce schéma appelé *méta-schéma* stocke le modèle d'ontologies dans un méta-modèle refléxif. Pour le schéma des ontologies, le méta-schéma joue le même rôle que celui joué par la métabase (ou catalogue système) dans les bases de données traditionnelles. En effet, le méta-schéma peut permettre : (1) un accès générique aux ontologies, (2) l'évolution du modèle d'ontologies utilisé, et (3) le stockage de différents modèles d'ontologies (OWL, DAML+OIL, PLIB, etc.). La figure 4 présente le méta-schéma de notre exemple.

Meta-Schema

Entity		
ID	**Name**	**SuperEntity**
Entity#1	Resource	
Entity#2	Class	Entity#1
Entity#3	Property	Entity#1

Attribute			
ID	**Name**	**Domain**	**Range**
Att#1	name	Entity#1	Type#1
Att#2	domain	Entity#3	Type#2
…	…	…	…

Type	
ID	**Name**
Type#1	String
Type#2	Entity#2

Figure 4: Méta-Schéma des BDBO de type 3

Ces trois catégories d'architectures de BDBO se comportent différemment selon le type d'information qui doit être géré. Dans la prochaine section, nous discutons de leur capacité de passage à l'échelle.

2.2 Scalabilité des BDBO

BDBO de type 1. L'approche utilisant une table verticale présente de sérieux problèmes de performance lorsque des requêtes requièrent de nombreuses auto-jointures sur cette table [3]. Pour obtenir de bonnes performances dans le traitement des requêtes, chaque colonne de la table verticale doit être indexée [13]. De plus, la colonne qui correspond au prédicat doit être clusterisée [2] ou des vues matérialisées doivent être créées [15]. Dans les deux cas, cette approche entraîne un coût de stockage supplémentaire et un surcoût de traitement des opérations de mises à jour. Et, même avec ces optimisations, plusieurs travaux ont montré que dans de multiple cas, les BDBO de type 2 surpassent les BDBO de type 1 [19, 13, 3].

BDBO de type 2. Les performances de ces BDBO dépendent de la représentation utilisée pour les données de niveau instance. Les évaluations de performance conduites dans [2, 19, 15, 1] ont montré que l'approche utilisant une table verticale pour les instances souffre des mêmes faiblesses que celles évoquées précédemment pour les BDBO de type 1. Ainsi, la représentation binaire a été considérée pendant longtemps comme la meilleure représentation pour les instances.

Cependant, les résultats expérimentaux obtenus sur la représentation table par classe ont remis en cause cette idée [9, 16]. Pour les requêtes où les classes à interroger sont spécifiées, la représentation table par classe surpasse l'approche binaire par un ratio souvent supérieur à 10, en particulier lorsque les instances sont associées à plusieurs propriétés [9]. De plus, les insertions et mises à jour sont plus rapides. Le seul cas où la représentation binaire est meilleure que la représentation table par classe correspond aux requêtes où la classe à interroger n'est pas spécifiée et qui ne concernent qu'un nombre faible de propriétés.

BDBO de type 3. L'ajout du méta-schéma dans les BDBO de type 3 n'améliore pas les performances des requêtes mais offre de nouvelles fonctionnalités comme nous l'avons précisé précédemment.

La disponibilité d'un méta-schéma d'ontologies dans la

base de données facilitera en particulier l'extension du modèle d'ontologies permettant d'implémenter l'approche d'étiquetage dynamique proposée dans cet article.

Cette étude sur la scalabilité des BDBO montre que, pour un nombre de cas d'utilisation correspondant aux différents benchmarks réalisés dans la littérature, les BDBO de type 2 ou 3 utilisant soit une représentation binaire, soit une représentation table par classe passent relativement bien à l'échelle et permettent la gestion de données du Web Sémantique de taille réelle. L'autre défi est de fournir en même temps des capacités de raisonnement.

2.3 Capacités de Déduction des BDBO

Les capacités de déduction ne sont pas le principal atout des bases de données. Pour réaliser des raisonnements, deux principales approches peuvent être suivies. La première approche consiste à réaliser le raisonnement *avant* le traitement des requêtes et à matérialiser tous les faits déduits et, en particulier, la fermeture transitive (FT) de toutes les relations transitives. Nous appelons cette approche *raisonnement à priori*. Cette approche permet le traitement efficace de requêtes puisque le raisonnement n'est pas réalisé à l'exécution. Son inconvénient est d'entraîner un coût de stockage supplémentaire et un surcoût de traitement des opérations de mises à jour. La seconde approche consiste à réaliser le raisonnement pendant le traitement des requêtes en produisant des faits déduits utilisés pour fournir le résultat des requêtes. Nous appelons cette approche *raisonnement à posteriori*. Cette approche est duale à la première : elle entraîne un surcoût pour le traitement des requêtes mais n'impose pas de coût de stockage et de mises à jour supplémentaire.

Actuellement, les BDBO supportent principalement les raisonnements sur la subsumption décrits dans [11] (c'est-à-dire, exploitant les relations `subClassOf` et `instanceOf`). La plupart des BDBO réalisent un raisonnement à posteriori en utilisant différents mécanismes tels que les vues [15], les techniques d'étiquetage [16] ou l'héritage de tables des bases de données relationnelles-objets [3, 5].

Certaines BDBO permettent des raisonnements plus complexes. Par exemple, ONTOMS supporte des raisonnements sur les instances utilisant les propriétés inverses, symétriques et transitives [16]. Mais, ces raisonnements sont en général réalisés sur des FT ce qui provoque un surcoût de stockage et de mise à jour important dans des applications de taille réelle. D'autres approches proposent d'utiliser des moteurs de règles des bases de données déductives (par exemple, un moteur de règles pour DATALOG) ou des raisonneurs OWL pour réaliser ces tâches de raisonnement plus complexes [14, 20, 4, 15]. Cependant, les bases de données déductives n'ont pas été largement adoptées en dehors du domaine académique et le temps de réponse de ce type d'architecture est souvent incompatible avec le besoin d'interraction homme-machine.

En fait, en plus de leur capacité à gérer des données de taille volumineuse, une des forces des bases de données est de permettre le traitement efficace de requêtes numériques ou alpha-numériques. Ainsi, des raisonnements à posteriori efficaces peuvent être réalisés si ces raisonnements déductifs peuvent être remplacés par des requêtes numériques (ou alpha-numériques). Les techniques d'étiquetage sont une application bien connue de cette approche [2, 7] où les relations transitives sont représentées soit par des intervalles numériques soit par des valeurs de type chaîne de caractères (ou vecteur de bits). Ces techniques sont très efficaces tant que la relation transitive définit une structure d'arbre. Lorsque c'est un DAG, les techniques d'étiquetage deviennent plus complexes et beaucoup moins efficaces. Dans la prochaine section, nous proposons une formalisation qui intègre les différentes techniques d'étiquetage et nous montrons que cette même formalisation peut être utilisée efficacement pour les structures de DAG rencontrées dans les applications spatiales et temporelles.

3. RAISONNEMENT NUMÉRIQUE SUR DES ENSEMBLES PARTIELLEMENT ORDONNÉS

3.1 Un Exemple Motivant le Problème

Le but du projet e-Wok Hub [2] est de gérer la mémoire de plusieurs projets d'ingénierie sur la capture et le stockage de CO_2. En particulier, un objectif important est d'améliorer la qualité des recherches de documents sur ce sujet. L'approche suivie consiste à utiliser des annotations de documents définies, autant que possible, de manière automatique. Comme exemple, nous nous sommes intéressés à l'aspect géographique des annotations. Une ontologie existante, appelée COG [3], qui décrit les zones géographiques françaises est utilisée pour annoter les documents. Dans cette ontologie, une *zone géographique* est représentée comme une instance d'ontologie caractérisée par un `nom`, un `type` (par exemple, pays, département ou ville) et des `frontières`. Les `frontières`, non représentées directement dans le COG, nous ont été fournies par le BRGM [4] dans le cadre du projet. De plus, les zones géographiques sont organisées dans une structure d'arbre en utilisant une relation transitive nommée `subdivision`. Cette relation a le sens suivant : `x subdivision y` $\Leftrightarrow$ `y` $\subset$ `x`. Ainsi, elle définit un ordre partiel sur les zones géographiques. De plus, cet ordre est arborescent, chaque zone n'est inclue que dans une zone de niveau supérieur.

La figure 5 présente un extrait de l'arbre des zones géographiques du COG. Chaque noeud représente une zone géographique et chaque arrête représente la relation de subdivision. La racine de l'arbre est le pays `France` qui est subdivisé en 2 départements : `Ile de France` et `Poitou Charentes`. Ce dernier département est lui-même subdivsé en 2 villes : `Poitiers` et `La Rochelle`.

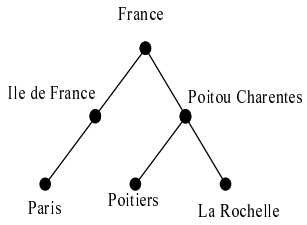

Figure 5: L'ontologie du COG : exemple de relation d'inclusion entre individus

Les documents sont automatiquement annotés en utili-

2. http://www-sop.inria.fr/acacia/project/ewok/index.html
3. Code Officiel Géographique, http://rdf.insee.fr/geo/
4. Bureau de recherches géologiques et minières (http://www.brgm.fr/)

sant le COG. Le prédicat d'annotation est nommé `geolocalized_in`. L'annotation (`doc geolocalized_in zone`) signifie que le document `doc` contient des informations à propos d'une partie ou de l'ensemble de la zone géographique `zone`. Nous notons que ce prédicat a un comportement particulier par rapport à l'ordre de `subdivision`. Si un document contient des informations à propos de `Poitiers`, il contient des informations à propos d'une partie du `Poitou Charentes`. Ainsi, (`doc geolocalized_in Poitiers`) implique (`doc geolocalized_in Poitou Charentes`). Notons que tous les prédicats dont le codomaine est une zone géographique n'ont pas forcément ce comportement. Par exemple, la personne qui dirige le `Poitou Charentes` ne dirige pas nécessairement la ville de `Poitiers`.

Ce comportement a une incidence sur les requêtes. En effet, si quelqu'un recherche tous les documents qui ont un rapport avec la zone géographique `zone`, le système doit raisonner sur la relation d'inclusion et retourner tous les documents annotés par les zones géographiques incluses dans `zone`. Des approches de raisonnement à priori ou à posteriori peuvent être utilisées pour fournir les bons résultats. Le raisonnement à priori consiste à stocker non seulement les annotations définies par les experts du domaine mais aussi les annotations qui peuvent être dérivées en utilisant la caractéristique de `geolocalized_in`. Compte tenu du nombre important de documents qui peuvent être gérés combiné au nombre de zones géographiques françaises (et du monde dans une seconde étape), le raisonnement à priori nécessite beaucoup d'espace de stockage et passerait difficilement à l'échelle. Une technique de raisonnement à posteriori naïve consisterait à calculer la FT de la relation d'inclusion en utilisant les opérateurs récursifs de SQL99 (si disponible dans le SGBD) ou des procédures stockées récursives. A nouveau, à cause de la quantité importante de documents combiné avec le nombre de zones géographiques, cette approche passerait difficilement à l'échelle en terme de temps de traitement des requêtes. La relation d'inclusion définissant une structure d'arbre, une technique d'étiquetage classique peut être utilisée pour représenter de manière compressée la FT de la relation d'ordre arborescent `subdivision`. Ces techniques consistent à assigner des valeurs à chaque noeud d'une hiérarchie selon la position du noeud. La figure 6 présente la mise en oeuvre de la technique *d'étiquetage par intervalles* sur l'exemple précédent du COG. Sur cet arbre, à chaque zone géographique est assignée une paire d'entiers, `bound1` et `bound2`, qui définissent un intervalle. Une zone `zone1` est une subdivision (récursive) de `zone2` si l'intervalle de `zone1` est inclus dans l'intervalle de `zone2`.

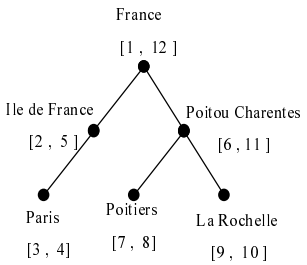

Figure 6: Exemple d'application d'une technique d'étiquetage à une structure d'arbre

Cette approche a été implémentée et passe parfaitement à l'échelle puisque les instances d'annotation sont relativement stables. En effet, une fois que toutes les instances du COG ont été insérées dans la base de données, la technique d'étiquetage ne nécessite aucun changement lorsque de nouvelles annotations de documents sont enregistrées.

Malheureusement, lorsque l'on annote automatiquement des documents, on ne rencontre pas seulement des noms de pays, départements ou villes. D'autres zones géographiques comme des régions, des districts ou des communes sont aussi utilisées. Notons qu'il existe toujours un ordre partiel entre toutes ces zones géographiques. Mais cet ordre ne définit plus un arbre. Il définit un DAG pour lequel la technique d'étiquetage par intervalles est beaucoup moins efficace. De plus, la plupart des techniques d'étiquetage nécessitent d'être recalculées lorsque les instances à indexer sont modifiées. Nous proposons dans la suite de cet article une formalisation permettant de sélectionner automatiquement la technique d'étiquetage à utiliser suivant le problème considéré et nous introduisons de nouvelles techniques d'étiquetage pour raisonner sur les zones spatiales et les périodes temporelles.

3.2 Formalisation Proposée

Pour commencer, caractérisons formellement le comportement des relations `geolocalized_in` et `subdivision`.

Soit $\mathtt{E}$ et $\mathtt{F}$ deux ensembles, $\mathcal{R} \subset \mathtt{E} \times \mathtt{F}$ et $\prec \subset \mathtt{F} \times \mathtt{F}$ deux relations binaires, avec $\prec$ étant une relation d'ordre, c'est-à-dire, réflexive, antisymétrique et transitive. Nous disons que $\mathcal{R}$ est *propagée par* l'ordre $\prec$ si et seulement si :

$$\forall \mathtt{x} \in \mathtt{E}, \forall \mathtt{y}, \mathtt{z} \in \mathtt{F},\ \mathtt{x}\ \mathcal{R}\ \mathtt{y} \wedge \mathtt{y} \prec \mathtt{z}\ \Rightarrow \mathtt{x}\ \mathcal{R}\ \mathtt{z}$$

et nous appelons *fermeture transitive propagée* (FTP) de $\mathcal{R}$ par $\prec$, notée $\mathcal{R}^+_\prec$:

$$\mathcal{R}^+_\prec = \{(\mathtt{x}, \mathtt{z}) \in \mathtt{E} \times \mathtt{F} \mid \exists \mathtt{y} \in \mathtt{F},\ \mathtt{x}\ \mathcal{R}\ \mathtt{y} \wedge \mathtt{y} \prec \mathtt{z}\}$$

Si $\mathcal{R} =$ `geolocalized_in` et $\prec =$ `subdivision`, pour une zone géographique donnée, $\mathcal{R}^+_\prec$ contient tous les documents qui sont annotés soit par cette zone, soit par une de ses subdivisions (récursives).

Raisonnement sur les Fermetures Transitives Propagées.

Nous notons que $\mathcal{R}^+_\prec$ est la composition de la fermeture transitive de $\prec$, notée $\prec^*$, avec $\mathcal{R}$: $\mathcal{R}^+_\prec = \prec^* \circ \mathcal{R}$. Et donc, une représentation efficace de la FT de $\prec$ permettrait d'obtenir une représentation efficace de la FTP de $\mathcal{R}$. Pour cela, nous utilisons une technique d'étiquetage. Une technique d'étiquetage $\mathcal{L}$ sur $(\mathtt{F}, \prec)$ est un triplet : $\mathcal{L} = (\mathtt{D}, \mathtt{label}, \mathtt{less\_or\_eq})$ où :

- $\mathtt{D}$ est un domaine concret ordonné $(\leq)$;
- $\mathtt{label} : \mathtt{F} \rightarrow \mathtt{D}$ est un morphisme d'ensembles ordonnés :
  $$\forall \mathtt{x}, \mathtt{y} \in \mathtt{F},\ \mathtt{x} \prec \mathtt{y} \Rightarrow \mathtt{label}(\mathtt{x}) \leq \mathtt{label}(\mathtt{y})$$
- $\mathtt{less\_or\_eq} : \mathtt{D} \times \mathtt{D} \rightarrow \mathtt{Boolean}$ est une fonction qui compare en temps constant deux valeurs de $\mathtt{D}$:
  $$\forall \mathtt{a}, \mathtt{b} \in \mathtt{D},\ \mathtt{less\_or\_eq}(\mathtt{a}, \mathtt{b}) \Leftrightarrow \mathtt{a} \leq \mathtt{b}$$

Ainsi, si pour tout $\mathtt{y} \in \mathtt{F}$, $\mathtt{label(y)}$ est pré-calculée et stockée dans la base de données et si `less_or_eq` peut être calculée en temps constant (par exemple, par comparaison numérique ou alphanumérique), le calcul de la FTP de $\mathcal{R}$ peut être fait en temps linéaire par un seul parcours de la relation $\mathcal{R}$.

Techniques d'Étiquetage Topologiques et Géométriques.

La plupart des techniques de labeling qui ont été proposées utilisent la structure topologique du treillis, qui représente

l'ordre sur l'espace F, pour définir les labels. Par exemple, comme nous l'avons vu précédemment, la technique d'étiquetage par intervalles proposée par Agrawal et al. [2], calcule l'intervalle numérique de chaque noeud en réalisant un parcours postfixe de l'arbre de recouvrement de la relation d'ordre entre toutes les instances connues. Dans la technique *par vecteur de bits*, proposée par Wirth [22], le label d'un noeud est représenté par un vecteur de n bits où n est le nombre d'instances de l'espace F. Un bit "1" à une position donnée identifie de manière unique un noeud dans le treillis et chaque noeud hérite des bits identifiant ses ancêtres dans le treillis. Ainsi, cet encodage est efficace tant qu'aucun changement majeur n'intervient dans la population de l'espace F, et, pour la technique par intervalles, tant que le treillis est un arbre. Lorsque des changements significatifs interviennent sur les instances de F, les labels doivent être recalculés.

En fait, lorsque l'on raisonne sur des domaines spatiaux ou temporels, l'espace sous-jacent a non seulement une structure topologique mais aussi une structure géométrique. Ainsi, il est associé à une métrique qui peut être utilisée pour définir les labels. Par exemple, dans la figure 5, les rectangles englobants, tout comme les les cercles englobants, pourraient être utilisés pour associer des labels aux zones géographiques. Lorsque l'on raisonne sur des périodes géologiques dans lesquelles plusieurs échelles de temps sont utilisées, une correspondance approximative entre chaque période géologique (exprimée en million d'années) et un temps géologique peut être établie. Notons que, contrairement aux labels basés sur la topologie, ces labels sont absolus. Ils représentent un savoir additionnel qui ne peut pas être automatiquement dérivé des instances connues de F ou des relations ; mais, ils n'ont pas besoin d'être changés quand le contenu de F est mis à jour. Tous ces divers labels peuvent être représentés comme des techniques d'étiquetage dans les BDBO permettant un raisonnement efficace sur les FT. Nous notons que les labels géométriques présentent deux différences avec les labels topologiques : (1) ils sont invariants pour une instance donnée quelles que soient les autres instances considérées, (2) ils ne peuvent pas être dérivés de propriétés non géométriques et non temporelles d'une instance d'ontologie. Ainsi, les propriétés spatiales et temporelles sont des propriétés ontologiques primitives d'objets spatiaux ou temporels. De ce point de vue, il est raisonnable de considérer que leurs valeurs, pour une instance donnée, peuvent être soit disponibles à un endroit donné (par exemple, par un service web) ou sont échangées avec la description de l'instance. Mon année de naissance tout comme la géolocalisation de Paris sont toutes les deux des propriétés ontologiques qui sont disponibles quelque part et qui peuvent être gérées dans des sources de données à base ontologique. Une difficulté est que les descriptions géométriques peuvent impliquer des structures de données complexes disponibles seulement dans des systèmes spécifiques (par exemple, SIG). En fait, d'importants raisonnements géométriques nécessitent seulement des données très simples. L'inclusion spatiale d'objets convexes peut être évaluée en utilisant les rectangles ou cercles englobants. La précédence temporelle nécessite seulement de comparer deux valeurs réelles ou deux intervalles. Ainsi, il est à la fois possible de restreindre l'ensemble des représentations géométriques autorisées et de supporter un large éventail de raisonnements (approximatifs) spatiaux et temporels. Notre suggestion est de supporter seulement des intervalles (une dimension (1D)), rectangles et cercles (deux dimensions (2D)).

4. CONCEPTION ET IMPLÉMENTATION

Cette section présente comment notre approche peut être implémentée dans différentes architectures de BDBO pour permettre l'automatisation du mécanisme de propagation de propriétés. Nous faisons maintenant l'hypothèse que E et F sont deux classes ontologiques. Pour représenter le fait qu'une propriété $\mathcal{R} : \mathtt{E} \times \mathtt{F}$ est propagée par un ordre partiel $\prec$ sur F, nous devons représenter : (1) le fait que $\prec$ est un ordre, (2) la technique d'étiquetage de cette ordre $\mathcal{L} = (\mathtt{D}, \mathtt{label}, \mathtt{less\_or\_eq})$, et (3) le fait que $\mathcal{R}$ doit être propagée par $\mathcal{L}$. Les langages (ou modèles) d'ontologies existants ne fournissent pas de primitive pour représenter ces trois informations. En conséquence, les langages d'ontologies ainsi que les BDBO doivent être étendus. Dans une base de données OWL par exemple, la première information nécessite d'ajouter une nouvelle valeur (nommée orderProperty) à l'ensemble des caractéristiques de propriétés OWL (transitiveProperty, symmetricProperty, etc.) puisque antisymmetric n'est disponible ni dans OWL1 ni dans OWL2. La troisième information nécessite d'ajouter une nouvelle valeur (nommée propagatedBy) à l'unique relation entre propriétés existante en OWL (inverseOf). Ainsi, ces deux informations nécessitent l'extension des langages d'ontologies. Par ailleurs, pour la seconde information, nous devons créer deux (méta-)tables additionnelles dans la BDBO. La première décrit les techniques d'étiquetage disponibles dans la BDBO. La seconde définit à quelle technique d'étiquetage est assignée un propriété d'ordre donnée. Ces deux tables sont des extensions des BDBO requises par notre modèle d'étiquetage. Dans la suite, nous présentons les grandes lignes du processus d'implémentation de notre approche et discutons des problèmes de représentation. Les différentes étapes de cette implémentation peuvent être réalisées à la fois sur les architectures de BDBO de type 2 et 3.

4.1 Extension de la Partie Modèle d'Ontologies des BDBO

Dans cette section, nous présentons les tables n-aires requises pour enregistrer les informations nécessaires. Les deux premières tables représentent les informations que nous proposons d'ajouter aux langages d'ontologies. Les deux dernières tables sont des tables systèmes.

- La table 1 property_characteristic contient les caractéristiques des propriétés. L'extension des langages d'ontologies consiste à permettre de représenter orderProperty comme une caractéristique.

Table 1: colonnes de la table property_characteristic

Colonne	Description
propertyId	référence l'unique identifiant de la propriété dans la table des propriétés.
characteristic	la caractéristique de la propriété (par exemple orderProperty, symmetricProperty, etc.)

- La table 2 property_to_property contient les relations entre deux propriétés. L'extension des langages d'ontologies consiste à permettre de représenter la relation propagated

by entre une propriété et une autre qui définit un ordre. Lorsqu'une technique d'étiquetage géométrique est utilisée et qu'un label est indiqué avec une donnée d'instance, par exemple comme un rectangle englobant, la relation d'inclusion peut souvent être implicite : elle doit être calculée par l'inclusion de formes géométriques définies par des labels géométriques. Dans ce cas, le `orderId` est remplacé par un mot réservé qui peut être `*geo_rectangle*`, `*geo_circle*`, `*geo_interval*`. Cela signifie que le `propertyId` est propagé par l'ordre d'inclusion des formes géométriques correspondantes. Les colonnes qui contiennent les labels géométriques sont nommées comme spécifié dans la table 3. Pour une relation `propagated by`, la colonne `direction` indique si la propagation est faite de manière *directe* (la même direction que la propriété d'ordre) ou de manière *inverse*. Par exemple, les lois applicables en Poitou-Charentes incluent celles définies dans des zones qui englobent le Poitou-Charentes ; ceci implique une propagation directe par rapport à l'ordre `subdivsion`, contrairement à la propriété `geolocalized in` où la propagation est faire de manière inverse. Par convention, l'ordre direct des relations géométriques est l'ordre croissant.

Table 2: colonnes de la table `property_to_property`

Colonne	Description
`PropertyId`	l'identifiant unique de la propriété propagée.
`orderId`	l'identifiant unique de la propriété dans la table `property_characteristic` ou un mot réservé (`*geo_rectange*`, ...)
`relationName`	le nom de la relation sémantique liant deux propriétés ; par exemple, `propagated by`, `inverse of`
`direction`	la direction de propagation par rapport à la relation d'ordre. Les valeurs autorisées sont `direct` et `reverse`.

- La table 3 `labeling_scheme` contient les informations à propos des différentes techniques d'étiquetage disponibles dans la BDBO.
Cette table est supposée être définie par l'administrateur de la base de données (DBA). Néanmoins, elle contient des lignes dont les 4 premiers attributs ont un contenu prédéfini. Ces lignes indiquent comment un étiquetage géométrique doit être spécifié pour être reconnu par le système. Ces lignes sont définies dans la table 4. Notons que, dans notre implémentation, ce type d'étiquetage géométrique est traité par l'extension PostGIS de PostgreSQL. Ceci permet donc à notre BDBO de supporter les ontologies géographiques comme définies par Cullot et al. [8] puisque cela permet de représenter des types géométriques et leurs fonctions de manipulation (et en particulier les types rectangle et cercle et les fonctions `less_or_eq` les concernant), de localiser les objets dans l'espace et de représenter des objets spatiaux. Notre proposition ne nécessite néanmoins pas du tout de disposer de toute la puissance d'un SIG pour mettre en oeuvre les mécanismes que nous proposons.
- La table 5 `property_schemes` contient des informations à propos des différentes techniques d'étiquetage associées à chaque propriété. Cette table est automatiquement générée par le système. Lorsqu'une propriété `propagated_by` est introduite dans la table 2, la technique d'étiquetage par défaut définie dans la table 3 est automatiquement implémentée si

Table 3: colonnes de la table `labeling_scheme`

Colonne	Description
`schemeId`	référence l'identifiant unique associée à la technique d'étiquetage
`numberOfColumns`	le nombre de colonnes utilisées pour représenter le domaine `D` (par exemple, 2 pour la technique d'étiquetage par intervalles)
`listColumnsSuffixes`	une liste de suffixes de colonnes utilisées pour représenter `D` (par exemple, { `bound1`, `bound2`})
`listColumnsTypes`	une liste de types de colonnes associés aux noms de colonnes dans `listColumnsSuffixes` (par exemple, {`int`, `int`})
`label`	le nom optionnel de la fonction SQL/PSM à utiliser pour calculer le label associé aux instances dont le label vaut `NULL`. Cette fonction est appelée sur `F` chaque fois qu'une ou plusieurs nouvelles instances de `F` sont ajoutées dans la base de données au sein d'une même transaction. Ce label n'existe pas (`NULL`) lorsqu'il doit être fourni de l'extérieur pour chaque instance (par exemple, pour des techniques d'étiquetage géométriques).
`less_or_eq`	le nom de la fonction SQL/PSM à utiliser pour évaluer si une instance est inférieure ou égale à un autre pour l'ordre défini par `propertyId`. Si $\mathcal{L}$ est la technique d'étiquetage par intervalles sur l'espace `F`, `i1` et `i2` sont deux instances, `i2` $\prec$ `i1` alors l'appel de fonction `less_or_eq(i2.bound1, i2.bound2,i1.bound1, i1.bound2)` retourne `true`.
`defaultScheme`	une valeur booléenne. La technique d'étiquetage par défaut associée à une nouvelle propriété définissant un ordre.

Table 4: techniques d'étiquetage prédéfinies dans la table `labeling-scheme`

schemeId	numberOfColumns	listColumnsSuffixes	listColumnsTypes	...
`*geo_interval*`	2	{bound1, bound2}	{float, float}	...
`*geo_rectangle*`	4	{xmin, xmax, ymin, ymax}	{float, float, float, float}	...
`*geo_circle*`	3	{xcenter, ycenter, radius}	{float, float, float}	...

le `orderId` identifie une propriété. Si le `orderId` est un mot réservé, le système vérifie seulement que la colonne d'étiquetage nécessaire est présente et une ligne dans la table 5 est aussi ajoutée. Le DBA peut changer la technique d'étiquetage par défaut si nécessaire.

Table 5: Colonnes de la table `property_schemes`

Colonne	Description
`propertyId`	l'identifiant unique de la propriété d'ordre dans la table `property_characteristic` ou de l'identifiant d'une propriété propagée lorsque son `orderId` est un mot réservé.
`schemeId`	l'identifiant unique de la technique d'étiquetage
`listProperties`	la liste des identifiants des propriétés associées à `listColumnsSuffixes` dans la classe `F`
`activeScheme`	une valeur booléenne. `true` si la technique d'étiquetage est activée.

4.2 Représentation des Instances

Les instances des classes de l'ontologie seront représentées selon la structure de données utilisée dans chaque BDBO. Pour automatiser la génération de propriétés spécifiques utilisées pour stocker les labels, elles sont gérées comme les autres propriétés et elles sont initialisées à `NULL` si aucune valeur n'est fournie.

4.3 Représentation des Annotations

Deux stratégies peuvent être utilisées pour représenter les annotations :

- utiliser une table binaire distincte pour chaque propriété d'annotation :
  - la première colonne `resourceId` référence l'unique identifiant d'une ressource ;
  - la seconde colonne `individuaId` référence l'unique identifiant d'une instance d'ontologie utilisée pour annoter la ressource.

  Si cette représentation est utilisée, une ou plusieurs jointures (ceci dépend la structure de la partie données) seront nécessaires pendant le traitement des requêtes pour récupérer la ou les valeurs de labels associées à la propriété définissant une relation d'ordre ;
- utiliser une vue matérialisée distincte pour chaque propriété d'annotation contenant les colonnes `resourceId` et `individualId` définies ci-dessus, et également pour toutes les colonnes de `listColumnsSuffixes` (représentation de la technique d'étiquetage). Notons que si des vues matérialisées sont utilisées, une politique de gestion des mises à jour doit être définie.

4.4 Traitement des Requêtes

Notre but est de réaliser un raisonnement numérique automatique pour les requêtes utilisant des propriétés qui ont des caractéristiques particulières comme une relation d'ordre ou une propagation par un ordre. Chaque requête doit être traitée par l'interpréteur d'une BDBO de la manière suivante :

- identification de la catégorie de la requête : chaque requête doit être analysée au niveau ontologique pour déterminer si elle implique une propriété présentant une caractéristique particulière. Ceci peut être réalisé en utilisant les tables `property_characteristic`, `property_to_property` et `property_schemes` ;
- interprétation de la requête : si la requête n'implique pas de propriétés particulières, elle subira le traitement habituel ; sinon, la requête sera d'abord transformée en une requête numérique en utilisant les informations contenues dans les tables `labeling_scheme` et `property_schemes`. Cette traduction dépend également de la représentation des annotations. La requête réécrite résultante sera ensuite exécutée efficacement par la base de données.

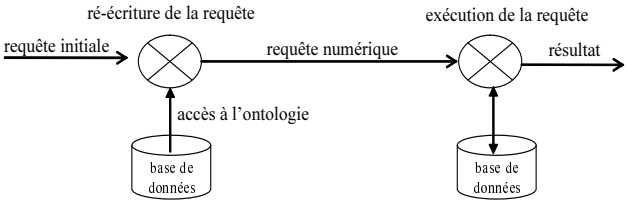

Figure 7: Étapes de traitement des requêtes.

La figure 7 illustre les différentes étapes suivies pour le traitement des requêtes. Dans ce qui suit, nous décrivons l'implémentation de notre approche que nous avons réalisée sur la BDBO OntoDB.

5. APPLICATION À L'ONTOLOGIE DU COG DANS LA BDBO ONTODB

Cette section décrit comment notre approche a été effectivement implémentée sur la BDBO OntoDB [9, 18] avec l'ontologie du COG en utilisant le langage OntoQL [12]. Jusqu'à présent, seules les techniques d'étiquetage par intervalles et par rectangles englobants ont été implémentées. Nous commençons par décrire brièvement la BDBO OntoDB.

5.1 OntoDB

OntoDB est une BDBO de type 3 conçue pour supporter l'évolution du schéma des ontologies et pour offrir un accès aux données au niveau ontologique. Actuellement, OntoDB est implémentée sur le SGBD PostgreSQL. Elle consiste en 4 parties. Les parties 1 et 2 sont les parties habituelles disponibles dans tous les SGBD, c'est à dire la partie données contenant les données d'instances et la partie métabase qui contient le catalogue système. La partie 3 (ontologie) et 4 (méta-schéma) sont spécifiques à OntoDB. Les données à base ontologique sont représentées dans OntoDB en utilisant une approche horizontale : une table est créée pour chaque classe de l'ontologie, ses colonnes correspondent au sous-ensemble des propriétés applicables de la classe, c'est-à-dire celles qui sont utilisées par au moins une instance de la classe. Cette représentation passe bien à l'échelle lorsque de nombreuses propriétés sont utilisées par instance [9].

5.2 Implémentation

L'ontologie COG est stockée dans la partie ontologie de OntoDB. La figure 8 présente le contenu des tables de la partie ontologie en utilisant un modèle d'ontologies simplifié. Ce modèle d'ontologies est stocké dans la partie méta-schéma. Ceci permet d'automatiser la génération de la structure de

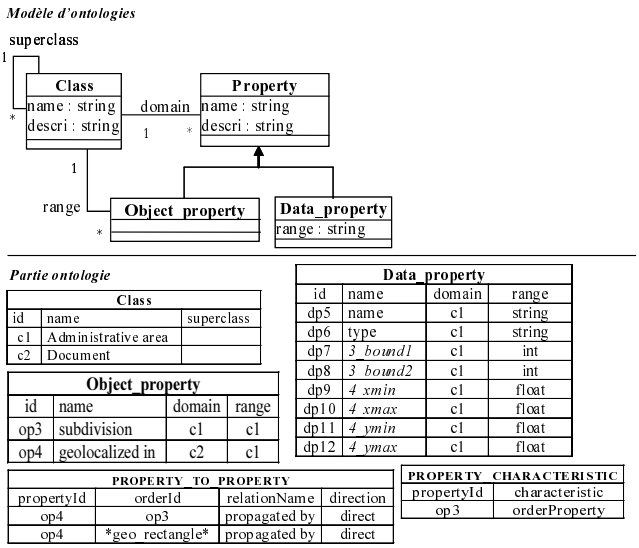

Figure 8: COG - OntoDB : Partie ontologie

LABELING_SCHEME						
schemeId	numberOf-Columns	listColumns-Suffixes	listColumns-Types	label	less_or_eq	default-Scheme
5	2	{bound1, bound2}	{int, int}	interval	include	TRUE
geo_rectangle	4	{xmin, xmax, ymin, ymax}	{float, float, float, float}	NULL	MBR-Contains	FALSE

PROPERTY_SCHEMES			
propId	schemeId	listProperties	activeScheme
3	5	{7,8}	TRUE
4	*geo_rectangle*	{9,10,11,12}	FALSE

Figure 9: COG - OntoDB : extension de la partie ontologie

la partie ontologie en utilisant une transformation de modèles. Lors la propriété `geolocalized_in` est insérée dans la table `property_to_property`, si le `orderId` référencé est `subdivision` et si la technique d'étiquetage par intervalles est activée par défaut, alors le système ajoute automatiquement les propriétés `bound1` et `bound2` (requises par la technique d'étiquetage par défaut) à son codomaine `spatial area`. Si la propriété `geolocalized_in` est à nouveau insérée dans la table `property_to_property` et si le `orderId` référencé est `*geo_rectangle*`, le système vérifie que les propriétés `xmin`, `xmax`, `ymin` et `ymax` sont présentes dans son codomaine `spatial area` et génère une nouvelle ligne dans la table `labeling scheme`. Nous avons étendu la partie ontologie de OntoDB avec les tables `labeling_scheme` et `property_schemes` décrites dans la section 4.1 et les avons peuplées avec les informations appropriées (voir figure 9).

Une fois que l'ontologie COG et les caractéristiques des propriétés et des relations ont été représentées, les instances doivent également être représentées dans la partie données en utilisant la structure de OntoDB (représentation table par classe). La représentation des instances est illustrée à la figure 10. Les valeurs de labels pour les instances peuvent soit être assignées en dehors de la BDBO, pour les techniques d'étiquetage prédéfinies, ou calculées automatiquement par la BDBO, pour les techniques d'étiquetage topologiques (par

subdivision	
Subdivision1	Subdivision2
France	Ile de france
France	Poitou-Charentes
Ile de france	Paris
Poitou-Charentes	La rochelle
Poitou-Charentes	Poitiers

spatial area					
id	name	type	...	bound1	bound2
11	France	country		1	12
12	Ile de France	department		2	5
13	Paris	town		3	4
14	Poitou-Charentes	department		6	11
15	Poitiers	town		7	8
16	La Rochelle	town		9	10

Document
id
21
22
23

Figure 10: COG - OntoDB : la partie données

geolocalized in	
RessourceId	IndividualName
21	Ile de france
22	Poitiers
23	La rochelle
24	Poitou Charentes

Table binaire

geolocalized in view			
RessourceId	IndividualName	bound1	bound2
21	Ile de France	2	5
22	Poitiers	7	8
23	La Rochelle	9	10
24	Poitou Charentes	6	11

Vue matérialisée

Figure 11: COG - OntoDB : représentation des annotations

exemple, pour la technique d'étiquetage 5, l'exécution de la fonction `interval` est automatiquement déclenchée par le système). OntoDB utilise l'identifiant (ID) des classes ontologiques pour générer le nom des tables et l'ID des propriétés pour générer le nom des colonnes dans la partie données. Ce mécanisme établit le lien entre la partie ontologie et la partie données. Pour une meilleure lisibilité, nous utilisons les noms (de classes, de propriétés, etc.) au lieu des ID. Actuellement, nous avons seulement implémenté les techniques d'étiquetage topologiques par intervalles numériques et les techniques d'étiquetage géométriques par rectangle englobant qui sont nécessaires pour la propriété `geolocalized_in` propagée par l'ordre `subdivision`. Les valeurs des coordonnées des rectangles englobants n'étant pas actuellement disponibles dans l'ontologie COG, nous les avons récupérées d'une autre source et entrées dans OntoDB.

Comme mentionné dans la section 4.3, deux stratégies peuvent être utilisées pour représenter les annotations. La figure 11 présente l'implémentation des de ces deux stratégies dans OntoDB. Dans la seconde stratégie, utilisant une vue matérialisée, seulement les valeurs de `bound1` et `bound2` des instances sont dupliquées pour chaque annotation parce que la technique d'étiquetage 1 (par intervalles) est la technique activée pour la propriété 3 (`subdivision`) (voir la table `property_schemes` dans la figure 9).

Nous pouvons maintenant nous intéresser à la transformation de requêtes pour qu'elles puissent être exécutées efficacement sur OntoDB en utilisant les capacités de raisonnement numérique de PostgreSQL. La figure 12 montre le traitement d'une requête SPARQL relativement simple utilisée dans le projet eWok-Hub. La requête à traiter est d'abord réécrite comme une requête OntoQL, un langage de requête pour les ontologies proche de SQL [12]. Ensuite la requête est classifiée en deux groupes suivant qu'elle nécessite ou non un traitement spécial. Cette classification est faite en utilisant la requête suivante qui permet de déterminer si la propriété `geolocalized in` (identifiant `4`) présente une caractéristique particulière :

```
SELECT p2p.relationName, p2p.direction,
       p2p.orderId, pc.characteristic
  FROM property_to_property p2p,
       property_characteristic pc
```

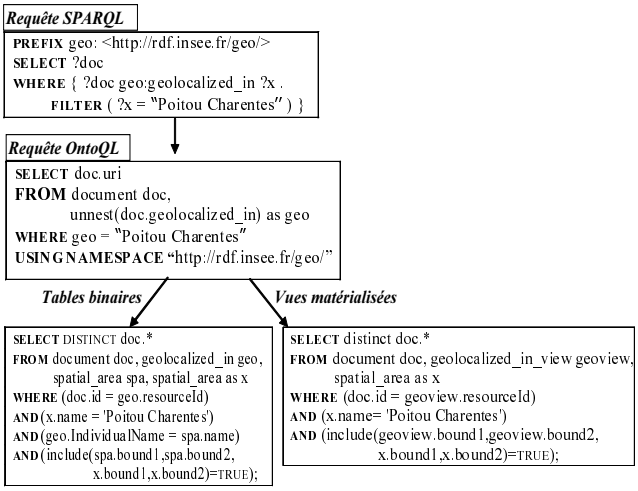

Figure 12: COG - OntoDB : exemple de réécriture d'une requête

```
WHERE p2p.propertyId = 4
  AND p2p.orderId = pc.propertyId
```

Le résultat de cette requête est le suivant :

relationName	direction	orderId	characteristic
propagated by	direct	3	orderProperty

Ce résultat indique que la propriété `geolocalized in` est propagée de manière directe par l'`object_property 3`, c'est-à-dire `subdivision`. En conséquence, la requête doit être réécrite pour qu'elle utilise la technique d'étiquetage par intervalles (technique associée à la propriété `geolocalized in` dans la table `property_schemes`). La réécriture suivant la représentation par tables binaires et par vues matérialisées est présentée sur la figure 12. Suivant les informations contenues dans la table `labeling_schemes`, ces réécritures utilisent la fonction `include` sur les colonnes suffixées par `bound_1` et `bound_2` pour déterminer si un document est annoté par une zone géographique incluse dans le `Poitou Charentes`.

6. CONCLUSION

La réalisation de la vision du Web Sémantique requiert des outils de gestion d'ontologies qui passent à l'échelle en terme de quantité de données gérées et qui réalisent des opérations de raisonnement sur les annotations basées sur des ontologies en un temps de réponse acceptable. Dans cet article, nous avons d'abord décrit l'état de l'art actuel des BDBO qui permettent de stocker à la fois les données de niveau ontologie et de niveau instance dans une base de données. Si certaines architectures passent relativement bien à l'échelle pour différentes applications, les capacités de raisonnement fournies sont principalement basées sur la représentation explicite de tous les faits qui peuvent être déduits par un raisonneur. Cette approche peut entraîner un fort surcoût de stockage pour des applications de taille réelle. Comme alternative, nous avons proposé une approche qui consiste à enrichir les instances d'ontologie utilisées comme annotations avec de nouvelles valeurs de propriétés de manière à pouvoir remplacer un raisonnement déductif par un traitement de requêtes numériques (ou alphanumériques).

Nous avons considéré deux types de raisonnement : les raisonnements par transitivité sur des ensembles partiellement ordonnés et des raisonnement sur la composition de deux propriétés, la seconde étant transitive. Ces cas englobent l'évaluation de propriétés liées à une taxonomie, les raisonnements de subsomption et les raisonnements d'inclusions spatiales et temporelles. Nous avons proposé une formalisation qui permet de caractériser ces cas au niveau des ontologies. Ceci permet au système d'une BDBO d'implémenter dynamiquement le raisonnement numérique lorsque de telles ontologies sont chargées. Cette formalisation nécessite trois types d'informations, chacune d'entres elles correspondant à une extension des modèles d'ontologies existants :

- le fait qu'une propriété définit un *ordre*, c'est-à-dire qu'elle soit transitive, réflexive et antisymétrique ou un *ordre arborescent*, c'est-à-dire qu'elle implique également l'unicité du majorant direct ;
- le fait qu'une propriété puisse être *propagée par* une autre propriété transitive ;
- une technique d'étiquetage permettant de spécifier le type d'étiquetage qui doit être utilisé pour remplacer le raisonnement déductif par un raisonnement numérique ou alphanumérique.

Deux types de techniques d'étiquetage existent. Les techniques d'étiquetage topologiques qui correspondent à différentes techniques proposées pour des structure d'arbre ou de DAG. Ce type de label peut être calculé par le système de la BDBO. Notre approche permet ainsi de spécifier de manière déclarative la technique d'étiquetage qui doit être utilisée pour une propriété donnée et de définir une technique d'étiquetage par défaut. Les techniques d'étiquetage géométriques sont utilisées pour le raisonnement spatial ou temporel. Les labels doivent être fournis par le système par l'intermédiaire de propriétés. Notre approche permet alors de spécifier les propriétés qui doivent être utilisées pour contenir les labels. Dans les deux cas, l'interpréteur de requêtes de la BDBO peut réécrire automatiquement les requêtes en utilisant les labels. Cette approche a été implémentée sur la BDBO OntoDB. Nous avons présenté les deux approches qui peuvent être utilisées pour réécrire les requêtes.

Remerciements. Les recherches décrites dans cet article ont été financées partiellement par le projet ANR eWok-Hub (05RNTL02706). Les auteurs remercient tous les membres du projet pour les discussions fructueuses. Un remerciement particulier à Eric Sardet, CRITT Informatique, qui a contribué à la conception et l'implémentation sur OntoDB.

7. REFERENCES

[1] D. J. Abadi, A. Marcus, S. R. Madden, and K. Hollenbach. Scalable Semantic Web Data Management Using Vertical Partitioning. In *Proceedings of the 33rd International Conference on Very Large Data Bases (VLDB'07)*, pages 411–422, 2007.

[2] R. Agrawal, A. Somani, and Y. Xu. Storage and Querying of E-Commerce Data. In *Proceedings of the 27th International Conference on Very Large Data Bases (VLDB'01)*, pages 149–158, 2001.

[3] S. Alexaki, V. Christophides, G. Karvounarakis, D. Plexousakis, and K. Tolle. The ICS-FORTH RDFSuite : Managing Voluminous RDF Description Bases. In *Proceedings of the 2nd International Workshop on the Semantic Web (SemWeb'01)*, pages 1–13, 2001.

[4] A. Borgida and R. J. Brachman. Loading data into description reasoners. *SIGMOD Record*, 22(2) :217–226, 1993.

[5] J. Broekstra, A. Kampman, and F. van Harmelen. Sesame : A Generic Architecture for Storing and Querying RDF and RDF Schema. In *Proceedings of the 1st International Semantic Web Conference (ISWC'02)*, pages 54–68, July 2002.

[6] E. I. Chong, S. Das, G. Eadon, and J. Srinivasan. An Efficient SQL-based RDF Querying Scheme. In *Proceedings of the 31st international conference on Very Large Data Bases (VLDB'05)*, pages 1216–1227, 2005.

[7] V. Christophides, D. Plexousakis, M. Scholl, and S. Tourtounis. On labeling schemes for the semantic web. In *Proceedings of the 12th International World Wide Web Conference (WWW'03)*, pages 544–555, 2003.

[8] N. Cullot, C. Parent, S. Spaccapietra, and C. Vangenot. Des SIG aux ontologies géographiques. *Revue internationale de géomatique*, 13(3/2003) :285–306, 2003.

[9] H. Dehainsala, G. Pierra, and L. Bellatreche. OntoDB : An Ontology-Based Database for Data Intensive Applications. In *Proceedings of the 12th International Conference on Database Systems for Advanced Applications (DASFAA'07)*, pages 497–508, 2007.

[10] S. Harris and N. Gibbins. 3store : Efficient Bulk RDF Storage. In *Proceedings of the 1st International Workshop on Practical and Scalable Semantic Systems (PSSS'03)*, pages 1–15, 2003.

[11] P. Hayes. *RDF Semantics*. World Wide Web Consortium, 2004. http://www.w3.org/TR/rdf-mt/.

[12] S. Jean, Y. Aït-Ameur, and G. Pierra. Querying Ontology Based Database Using OntoQL (an Ontology Query Language). In *Proceedings of On the Move to Meaningful Internet Systems 2006 : CoopIS, DOA, GADA, and ODBASE, OTM Confederated International Conferences (ODBASE'06)*, volume 4275 of *Lecture Notes in Computer Science*, pages 704–721. Springer, 2006.

[13] L. Ma, Z. Su, Y. Pan, L. Zhang, and T. Liu. RStar : an RDF Storage and Query System for Enterprise Resource Management. In *Proceedings of the 30th International Conference on Information and Knowledge Management (CIKM'04)*, pages 484–491, 2004.

[14] J. Mei, L. Ma, and Y. Pan. Ontology query answering on databases. In *Proceedings of the 5th International Semantic Web Conference (ISWC'06)*, pages 445–458, 2006.

[15] Z. Pan and J. Heflin. DLDB : Extending Relational Databases to Support Semantic Web Queries. In *Proceedings of the 1st International Workshop on Practical and Scalable Semantic Systems (PSSS'03)*, pages 109–113, 2003.

[16] M. J. Park, J. H. Lee, C. H. Lee, J. Lin, O. Serres, and C. W. Chung. An efficient and scalable management of ontology. In *Proceedings of the 12th International Conference on Database Systems for Advanced Applications (DASFAA'07)*, pages 975–980, 2007.

[17] J. Petrini and T. Risch. SWARD : Semantic Web Abridged Relational Databases. In *Proceedings of the 18th International Conference on Database and Expert Systems Applications (DEXA'07)*, pages 455–459, 2007.

[18] G. Pierra, H. Dehainsala, Y. Aït-Ameur, and L. Bellatreche. Base de Données à Base Ontologique : principes et mise en œuvre. *Ingénierie des Systèmes d'Information*, 10(2) :91–115, 2005.

[19] Y. Theoharis, V. Christophides, and G. Karvounarakis. Benchmarking Database Representations of RDF/S Stores. In *Proceedings of the 4th International Semantic Web Conference (ISWC'05)*, pages 685–701, 2005.

[20] R. Volz, S. Staab, and B. Motik. Incrementally Maintaining Materializations of Ontologies Stored in Logic Databases. *Journal of Data Semantics II*, 3360 :1–34, 2005.

[21] K. Wilkinson, C. Sayers, H. Kuno, and D. Reynolds. Efficient RDF Storage and Retrieval in Jena2. In *Proceedings of the 1st International Workshop on Semantic Web and Databases (SWDB'03)*, pages 131–150, 2003.

[22] N. Wirth. Type extensions. *ACM Transactions on Programming Languages and Systems (TOPLAS)*, 10(2) :204–214, 1988.