Etude Comparative des Systèmes de Bases de Données à base Ontologiques

Bery Mbaiossoum[*,], Selma Khouri[*,***], Ladjel Bellatreche [*], Stéphane Jean [*], Mickael Baron[*]**

**LIAS/ENSMA - Université de Poitiers 86960 Futuroscope Cedex France*
***University of N'Djamena, Chad*
****National High School for Computer Science, Algeria*
{mbaiossb, selma.khouri, bellatreche, jean, baron}@ensma.fr

RÉSUMÉ. *Nous assistons à une importante émergence des ontologies de domaine dans le monde académique et industriel. Cette adoption a fait naître de nouveaux besoins concernant la gestion des instances ontologiques. Plusieurs formalismes ontologiques ont été développés, ce qui rend le partage des instances ontologiques difficile. Des solutions de persistance ont été proposées pour stocker ces instances via des SGBD, ce qui fait naître la notion de bases de données à base ontologique (*BDBO*). Chaque SGBD à une* BDBO *possède sa propre architecture et son modèle de stockage pour la partie ontologie et instances. Dans cet article, nous présentons un état des lieux sur les ontologies, leurs formalismes, les modèles de stockage et les architectures cibles de leurs SGBD. Ensuite, une formalisation des* BDBO. *Finalement, une étude de performance en utilisant le banc d'essai Lehigh University BenchMark de trois SGBD dont deux issus du monde industriel (Oracle et IBM SOR) et un issu du milieu académique appelé OntoDB du LIAS-ENSMA.*

ABSTRACT. *Nowadays, domain ontologies are largely advocated by industrial and academic communities. This adoption gives raise to new needs for managing large amounts of ontological instances. Several ontological formalisms were proposed. Persistency solutions were developed for storing these instances in DBMS. Note that each DBMS has its own architecture and storage model for instances and ontologies. In this paper, we first present a state of art on ontologies, their formalisms, storage models and architectures of their target DBMS. Secondly, a formalisation of ontologies and ontology-based databases is given. An experimental study using Lehigh University BenchMark is conducted to prove three DBMS: two from industrial community (Oracle and IBM SOR) and one from academic research laboratory (LIAS-ENSMA).*

MOTS-CLÉS : *Ontologies, BDBO, Web sémantique, évaluation des performances*

KEYWORDS: *Ontology, OBDB, Semantic Web, performance evaluation*

1. Introduction

Les ontologies sont des conceptualisations qui permettent de représenter explicitement la sémantique d'un domaine par des modèles objets consensuels dont chaque concept (classe ou propriété) est associé à un identificateur universel permettant de référencer la sémantique qui lui correspond. Selon Gruber (Gruber, 1993), une ontologie est une spécification explicite d'une conceptualisation. Cette définition a été enrichie par Jean et al. (Jean *et al.*, 2007) en décrivant une ontologie comme une représentation formelle, explicite, référençable et consensuelle de l'ensemble des concepts partagés d'un domaine en termes de classes d'appartenance et de propriétés caractéristiques. Cette définition met en avant trois caractéristiques qui distinguent une ontologie de domaine des autres modèles informatiques tels que les modèles conceptuels et les modèles de connaissance. Une ontologie est une représentation :

– *formelle* : exprimée dans un langage de syntaxe et de sémantique formalisé (RDF Schéma (Brickley *et al.*, 2002), DAML+OIL (Connolly *et al.*, 2001), OWL(Bechhofer *et al.*, 2004), PLIB(ISO-13584-42, 1998), etc.) permettant ainsi des raisonnements automatiques ayant pour objet soit d'effectuer des vérifications de consistance, soit d'inférer de nouveaux faits ;

– *consensuelle* : admise par l'ensemble des membres (et des systèmes) d'une communauté et,

– *référençable* : toute entité ou relation décrite dans l'ontologie peut être directement référencée par un symbole (« identifiant »), à partir de n'importe quel contexte, afin d'expliciter la sémantique de l'élément référençant.

Les ontologies sont aujourd'hui utilisées dans un nombre croissant d'applications et de domaines variés. Par exemple, l'annotation de documents par des instances ontologiques est utilisée pour faciliter la recherche d'information dans le domaine du Web. Dans le domaine technique, les ontologies ont été utilisées pour représenter des catalogues de composants industriels. Enfin, dans les applications d'intégration de données, d'intégration des applications et l'intégration des plateformes, les ontologies offrent des solutions pour réduire les conflits syntaxiques et sémantiques véhiculées par les données, les applications et les plateformes. Ce développement continu de la notion d'ontologie a également été montré récemment par la communauté de modélisation conceptuelle (*Conférence Entity Relationhip* (ER'2011) à Bruxelles) qui a été amenée à proposer le concept d'une ontologie universelle[1]. L'utilisation massive de la notion d'ontologie par plusieurs communautés (linguistique, intelligence artificielle, base de données, etc.) a entrainé la génération d'une masse importante d'instances ontologiques. Par instance ontologique, nous entendons un objet dont le sens est défini par son appartenance à une classe ontologique et par les valeurs d'un certains nombre de propriétés définies dans la même ontologie. Nous appellerons données à base ontologiques (DBO) un ensemble d'instances ontologiques.

1. http ://blogs.euranova.eu/ ?p=943

L'utilisation croissante des ontologies a également eu un autre impact : le développement de nombreux formalismes d'ontologie : RDF (Klyne *et al.*, 2004), RDFS (Brickley *et al.*, 2002), OWL (Bechhofer *et al.*, 2004), PLIB (Pierra, 2008), FLIGHT (Bruijn *et al.*, 2004), etc. Chacun de ces formalismes cible un domaine d'application particulier et introduit, pour ce faire, des primitives de modélisation particulières. Par exemple, le formalisme PLIB est largement utilisé dans le domaine de l'ingénierie (Bellatreche *et al.*, 2006) tandisque les formalismes RDFS et OWL sont massivement utilisés dans le domaine du Web Sémantique.

Par ailleurs, les BDBO étaient initialement gérées par des outils en mémoire centrale(le cas de *Corese* : http ://www-sop.inria.fr/edelweiss/software/corese/). Avec la multiplication des ontologies de domaine, et le volume important de données à manipuler, est apparu le besoin de systèmes susceptibles de gérer des ensembles de données à base ontologique de grande taille. De tels systèmes sont appelés des systèmes de gestion de bases de données à base ontologique (BDBO). Différentes BDBO ont ainsi été proposées dans le monde académique telles que OntoDB (Dehainsala *et al.*, 2007), Ontobroker (Fensel *et al.*, 1998), RDFSuite (Alexaki *et al.*, 2001), SESAME (Broekstra *et al.*, 2002) et OntoMS (Park *et al.*, 2007) ainsi que dans le monde industriel comme par exemple Oracle (Murray, 2008.) ou IBM SOR (Scalable Ontology Repository) (Lu *et al.*, 2007). Comparé au développement des bases de données traditionnelles, ces systèmes ont été développés et proposés dans un espace de temps très court. Le développement de nombreuses BDBO résulte principalement de (1) *la diversité des formalismes* : chaque BDBO utilise un formalisme particulier pour définir ses ontologies (OWL, PLIB ou FLIGHT), (2) *la diversité des modèles de stockage* : contrairement aux bases de données traditionnelles, où le modèle logique est stocké selon une approche relationnelle, dans une BDBO, une variété de modèles de stockage (représentation horizontale, spécifique, etc.) sont utilisés pour stocker deux niveaux de modélisation : le niveau ontologie et le niveau des instances ontologiques et (3) *la diversité des architectures cibles* utilisées par le système de gestion de bases de données : une BDBO peut utiliser un seul ou plusieurs schémas de base de données pour stocker l'ensemble des données.

La diversité des BDBO proposées rend leur comparaison difficile. Actuellement, la plupart des comparaisons effectuées, s'intéresse uniquement aux performances de ces BDBO sans prendre en compte ni leur possibilité de modélisation ni l'impact de leur architecture sur les performances offertes. Aussi, nous proposons dans cet article une formalisation de la notion de BDBO afin de fournir un cadre formel pour leur comparaison ainsi que pour clarifier la diversité des BDBO existantes. Nous montrons que cette formalisation permet de représenter une variété de BDBO en comparant trois BDBO dont deux issus du monde industriel (Oracle et IBM SOR) et une du monde académique (OntoDB). La formalisation proposée permet de comparer l'architecture et les modèles de stockage de ces BDBO. Pour compléter cette proposition nous proposons une comparaison des performances de ces BDBO selon deux critères : le temps de chargement et le temps de traitement de requêtes. Cette comparaison est réalisée en utilisant le banc d'essai LUBM (Lehigh University Benchmark) : http ://swat.cse.lehigh.edu/projects/lubm/.

Ce papier est divisé en sept sections : la section 2 présente les concepts de base liés aux ontologies, leurs formalismes et les BDBO. La section 3 présente les modèles de stockage dédiés aux BDBO (concernant à la fois les instances et les ontologies) et les architectures des SGBD cibles des BDBO. Section 4 décrit une formalisation des BDBO ainsi que les trois BDBO tests : Oracle, IBM et OntoDB. La section 5 montre les développements réalisés pour pouvoir tester la BDBO OntoDB avec une ontologie OWL. La section 6 montre les résultats expérimentaux obtenus en utilisant le banc d'essai LUBM. Enfin, la section 7 conclut le papier.

2. Notions sur les ontologies : concepts et formalismes

Les ontologies de domaine peuvent se définir comme une conceptualisation d'un domaine en termes de concepts et de propriétés. Gruber (Gruber, 1993) distingue deux types de concepts dans une ontologie conceptuelle : les concepts primitifs et les concepts définis. Les concepts primitifs ou canoniques représentent des concepts ne pouvant être définis par une définition axiomatique complète. Les concepts primitifs sont une base sur laquelle peuvent être définis d'autres concepts appelés concepts définis ou concepts non canoniques.
Plusieurs formalismes ont été proposés pour concevoir des ontologies qui diffèrent selon l'objectif de la conception. Certains formalismes ont pour but la description d'ontologies en vue de la gestion et l'échange des données. Ces formalismes cherchent à définir la sémantique des concepts de manière unique et précise et utilisent donc des concepts canoniques. Les modèles RDF-Schema et PLIB sont des exemples de formalismes permettant de décrire de telles ontologies. RDF-Schéma est un modèle issu du web sémantique, étendant le modèle RDF par des constructeurs permettant la définition de classes et de propriétés. Le modèle PLIB (Parts LIBrary) (Pierra, 2008) est un modèle défini dans le cadre du projet PLIB pour la description des différentes catégories de composants industriels et de leurs instances. Dans cette description, les propriétés jouent un rôle essentiel. Des classes sont définies seulement pour représenter des domaines de propriétés, et chaque propriété est définie dans le domaine d'une classe et n'a de sémantique que pour cette classe et ses sous classes. D'autres formalismes ont été proposés pour la description d'ontologies permettant de définir des correspondances entre vocabulaires et offrant ainsi des possibilités de déduction et d'inférence. Ces formalismes définissent ainsi des concepts canoniques et non canoniques. Les modèles du web sémantique comme *Daml*, *OIL*, *Daml+OIL* et *OWL* présentent des langages d'inférence. Ces modèles étendent le modèle *RDF-Schema* par des opérateurs permettant la définition d'ontologies conceptuelles plus expressives et offrant des capacités de raisonnement. Le modèle OWL est actuellement reconnu comme le standard pour représenter des ontologies dans le web. OWL se décline en trois versions : *OWL-Lite, OWL-DL* et *OWL-Full*, qui sont des sous-modèles présentant un compromis entre leur pouvoir expressif et leur décidabilité de raisonnement. *OWL-Full* offre une grande expressivité mais ses raisonnements ne sont pas décidables, contrairement à *OWL-Lite* et DL.

3. Stockage des ontologies dans des bases de données

Comme nous l'avons indiqué précédemment, la nécessité de stocker les ontologies et leurs instances au sein d'une base de données résulte du volume de données ontologiques devenant de plus en plus important. Plusieurs BDBO ont été proposées. Ces BDBO doivent permettre le stockage des ontologies et de leurs instances selon un format donné. Différents schémas de stockage ont été utilisés. Le modèle ontologique et les instances ontologiques peuvent être stockés conjointement utilisant le même format de stockage ou séparément utilisant des formats différents. Le stockage de ces modèles donne lieu à différentes architectures. Nous détaillons dans ce qui suit les modèles de stockage et les architectures des BDBO.

3.1. *Les schémas de stockage ontologiques*

Trois principales approches sont utilisées pour la représentation des ontologies au sein des bases de données (Fankam, 2009) :

– *Approche verticale :* consiste à représenter l'ontologie par une table à trois colonnes. Ces colonnes représentent respectivement : (1) l'identifiant de la ressource ontologique (classe, propriété ou instance ontologique), (2) le nom de la ressource, et (3) la valeur de cette ressource. Cette représentation facilite l'insertion de nouveaux triplets. Son interrogation est complexe car elle peut nécessiter plusieurs opérations d'auto-jointure. La BDBO Sesame et celle d'oracle utilise cette représentation pour la représentation des instances ontologiques.

– *Approche binaire* : elle consiste à décomposer les relations en deux catégories : relations unaires (pour l'appartenance aux classes), et relations binaires (pour les valeurs de propriétés). L'approche binaire se décline en trois variantes selon l'approche adoptée pour la représentation de l'héritage : (1) une table unique pour toutes les classes de l'ontologie, (2) une table par classe avec héritage de table (si un SGBD relationnel objet est utilisé) et (3) une table par classe sans héritage de table. La BDBO SOR utilise cette approche binaire pour la représentation des ontologies.

– *Approche horizontale* : cette approche est similaire à la représentation traditionnelle utilisée par les SGBD relationnels. Elle consiste à associer à chaque classe ontologique une table ayant une colonne pour chaque propriété associée à une valeur pour au moins une instance de cette classe. La BDBO OntoDB utilise l'approche horizontale pour la représentation de son modèle ontologique et également de ces instances.

NB : Il existe des BDBO qui combinent ces approches, on parle d'*approche hybride*.

3.2. *Les architectures des Bases de Données à Base Ontologique (BDBO)*

Les premières solutions de BDBO ont stocké les instances ontologiques d'une façon similaire aux bases de données classiques en utilisant deux parties : les données et le catalogue système. La partie "données" dans cette structure représente les instances

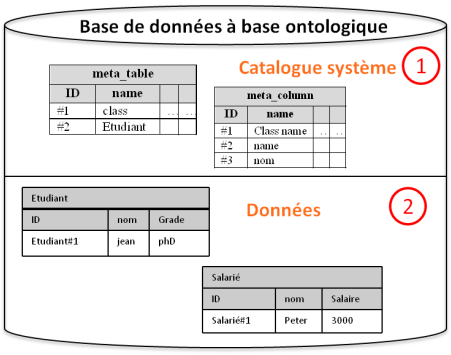

Figure 1. *Architecture deux quarts*

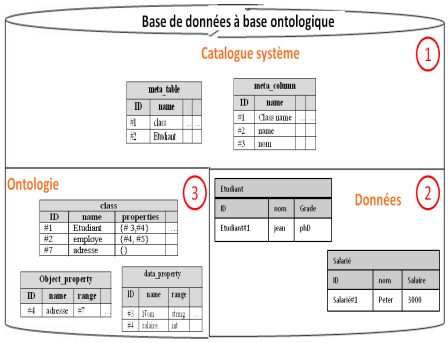

Figure 2. *Architecture trois quarts*

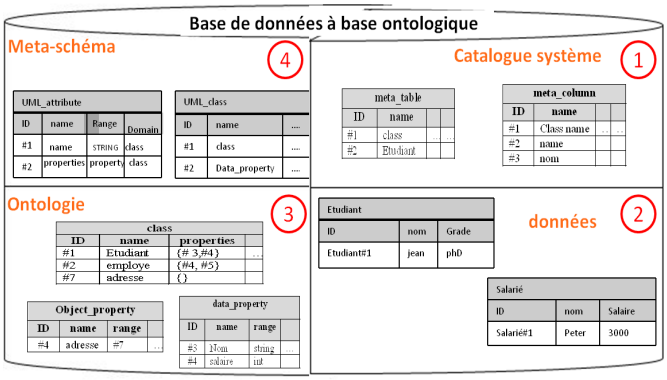

Figure 3. *Architecture quatre quarts.*

ontologiques mais également le schéma de l'ontologie (classes, propriétés, etc.). Cette architecture, présentée sur la figure 1 que nous appelons de type I ou architecture "deux quarts", impose ainsi le même schéma de stockage pour le schéma ontologique et pour les instances ontologique. Jena utilise une architecture de ce type qui stocke l'ontologie (modèle et instances) sous forme de triplet RDF (Sujet, Prédicat, Objet). Cette architecture ne permet pas de dissocier le schéma de l'ontologie des instances ontologiques.

Pour pallier cet inconvénient, une deuxième architecture de type II ou architecture "trois quarts" (figure 2) a été proposée où les instances ontologiques et le modèle ontologique sont stockés en deux schémas différents. Cette nouvelle architecture scinde la base en trois parties : catalogue système, schéma du modèle ontologique, schéma des instances ontologiques. IBM SOR (Lu *et al.*, 2007) est un exemple de BDBO suivant cette deuxième architecture. Même si les deux schémas de stockage (modèle et ins-

tances) sont indépendants, cette architecture manque de flexibilité dans la mesure où elle impose un schéma d'ontologie figé ce qui ne permet pas d'introduire de nouveaux concepts issus d'autres modèles d'ontologies.

Une troisième architecture de type III (architecture "quatre quarts") a été proposée dans le cadre du projet OntoDB (OntoDB1 et OntoDB2) visant à répondre au besoin de flexibilité du modèle d'ontologie utilisé dans la BDBO. Cette architecture propose pour cela une partie nommée méta-schéma qui joue pour les ontologies le même rôle que celui joué par le système catalogue pour les données. Cette partie permet un accès générique au modèle ontologique ainsi que l'introduction de nouveaux constructeurs issus de différents formalismes ontologiques. Cette vision qui permet de voir l'évolution des BDBO en termes de schémas de stockage et d'architectures est illustrée dans la figure 3.

4. Formalisation des BDBO

Pour représenter la diversité des structures de BDBO, il est nécessaire de proposer une structure générique permettant de représenter cette diversité en terme de : modèle ontologique, instances ontologiques, schéma de stockage du modèle ontologique, schéma de stockage des instances ontologiques et architecture de BDBO . Nous proposons la formalisation suivante :
$BDBO :< MO, I, Sch, Pop, SM_{MO}, SM_{Inst}, Ar >$, où

– MO : représente un modèle ontologique générique. Ce modèle est formalisé par le 5-uplet suivant : <C, P, Applic, Ref, Formalisme>

- C représente les classes du modèle ontologique.

- P représente les propriétés du modèle ontologique.

- $Applic : C \rightarrow 2^P$ est une fonction qui permet de lier chaque classe aux propriétés qui lui sont attachées.

- $Ref : C \rightarrow (operateur, Exp(C))$ est une fonction qui associe à chaque classe un opérateur (d'inclusion ou d'équivalence) et une expression sur d'autres classes. Les expressions définies pour les ontologies OWL basées sur les logiques de description présentent à notre point de vue un ensemble d'opérateurs complet couvrant plusieurs formalismes ontologiques. Ces expressions utilisent les opérateurs suivants : opérateurs ensemblistes (intersectionOf $(\cap)$, unionOf $(\cup)$, complementOf $(\neg)$), restrictions de propriétés (AllValuesFrom $\forall p.C$, SomeValuesFrom $\exists p.C$, HasValue $\ni p.C$) et les opérateurs de cardinalités ($\geq nR.C$, $\leq nR.C$). Exp peut être la fonction d'identité qui associe à une classe la même classe (la classe se définit par elle-même comme la plus grande classe de la hiérarchie "Thing").

- Formalisme est comme son nom l'indique le formalisme du modèle ontologique adopté.

Par exemple, une ontologie PLIB sera définie comme une instance du 5-uplets suivant : <Classes, Propriétés, Applic, <Opérateur de subsomption(Sub)>, PLIB>. L'opérateur

OntoSub de Plib est un opérateur permettant de définir un héritage partiel, où une classe référence une autre classe en héritant de tout ou d'une partie de ses propriétés. Une ontologie OWL sera définie par : <Classes, Propriétés, Applic, Opérateurs de logique de description, OWL>

– I : représente l'ensemble des instances ontologiques

– $Sch : C \rightarrow 2^P$ est une fonction qui associe à chaque classe l'ensemble des propriétés pour lesquelles les instances de cette classe sont valuées.

– $Pop : E \rightarrow 2^I$, est une fonction qui associe à chaque classe ses instances de l'ensemble I.

– Modèle stockage (SM_{MO}) : le schéma de stockage du modèle ontologique (vertical, horizontal, etc)

– Modèle stockage (SM_{Inst}) : le schéma de stockage des instances ontologiques.

– Modèle d'architecture (Ar) : le type d'architecture de la base (Type I, II ou III).

Selon cette formalisation, la BDBO d'Oracle, d'IBM SOR et d'OntoDB sont respectivement représentées par :
$BDBO_{Oracle} :< MO :$<Classes, Propriétés, Applic, Opérateurs (RDFS, OWLSIF et OWLPrime), (RDFS, OWLSIF ou OWLPrime selon les versions)>, Instances RDF, ϕ, Tables tables RDF_link and RDF_values donnant les instances de chaque classe, Vertical, Vertical, Type I>,
$BDBO_{IBM} :< MO :$<Classes, Propriétés, Applic (propriétés de chaque classe), opérateurs de logique de description, OWL>, Instances Owl, Applic, Instances de chaque classe, Horizontal, Binaire, Type II>.
$BDBO_{OntoDB} :< MO :$<Classes, Propriétés, Applic (propriétés de chaque classe), opérateur d'héritage et opérateur OntoSub, PLIB>, Instances plib, Sch $\subseteq$ Applic, Instances de chaque classe, Horizontal, Horizontal, Type III>.

4.1. *Présentation des* BDBO *utilisées*

Nous nous intéressons à trois BDBO dont l'une est issue du monde de la recherche (OntoDB) et les deux autres issues du monde industriel (oracle et SOR d'IBM).

4.1.1. *Base de données sémantique OntoDB*

OntoDB (Dehainsala *et al.*, 2007) est une architecture de base de données à base ontologique conçue par le Laboratoire d'Informatique et d'Automatique des Systèmes (LIAS). Une première monture est implantée sur le SGBD/RO PostGreSQL. Il offre une nouvelle façon de concevoir des applications de bases de données en stockant dans la base de données explicitement les données, le modèle conceptuel qui définit la structure des données et l'ontologie qui définit le sens de ces données. OntoDB supporte l'évolution du schéma des ontologies et offre un accès aux données au niveau ontologique. Les ontologies gérées jusque-là sous OntoDB, sont conformes au modèle d'ontologies PLIB. Cependant, OntoDB est prévue pour supporter n'importe quel type

d'ontologie. En effet, l'architecture d'OntoDB présentée sur la figure 4, dispose d'un méta-schéma susceptible de prendre en compte tout schéma d'ontologie. Cela permet de manipuler aussi les ontologies RDF(S), OWL et autres.
OntoDB utilise une architecture de type III ("quatre quarts") et un modèle de stockage horizontal : une table est créée pour chaque classe de l'ontologie, ses colonnes correspondent au sous-ensemble de propriétés applicables de la classe. Il est doté d'un langage de requête : OntoQL (Jean *et al.*, 2006).

OntoDB a été largement utilisé dans des projets industriels et ANR (Projet ANR DaFOE4App : Differential And Formal Ontologies Editor For Applications et ANR E-wok hub : http ://www-sop.inria.fr/edelweiss/projects/ewok/, etc.). Cette utilisation a été souvent basée sur les données techniques qui sont représentées par le formalisme PLIB. Les deux autres BDBO étudiés supportent le modèle OWL. Aussi dans cet article, nous allons examiner la capacité d'OntoDB à supporter le modèle OWL.

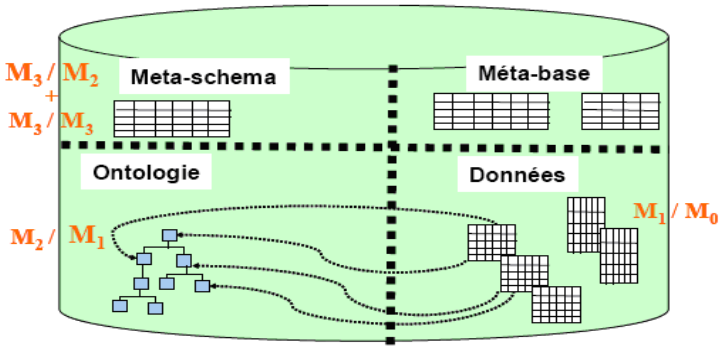

Figure 4. *Architecture d'OntoDB*

4.1.2. *La Base de données sémantique d'Oracle*

Oracle a incorporé des supports pour les langages RDF et OWL dans son système pour permettre à ses clients de bénéficier d'une plateforme de gestion de données sémantiques. Cette fonctionnalité a été implantée pour la première fois dans le SGBD Spatial Oracle 10g et est désormais en option dans les bases de données Oracle.
Oracle a défini deux sous-classes d'OWL DL qu'il exploite dans les processus d'inférence (Wu *et al.*, 2008) : OWLPrime et OWLSIF.
Oracle utilise une architecture de type I ("deux quarts") et un modèle de stockage en triplet. La table de triplet utilisée a également été améliorée par une décomposition (normalisation) (Wu *et al.*, 2008). Ainsi, pour éviter de manipuler les longues chaines de caractères (les URIs), les identificateurs sont générés pour chacune d'elles. Les valeurs lexicales des sujets, prédicats et objets sont mappées en identificateur (ID) entiers générés par le système. Oracle offre également la possibilité de faire des déductions de connaissances en se basant sur les règles d'inférence définies pour le modèle OWL.

4.1.3. *La Base de données sémantique d'IBM : SOR*

La société IBM s'est investie dans plusieurs travaux sur les technologies d'intégration et de gestion des données sémantiques et surtout celles relatives au Web sémantique. Elle s'est impliquée dans le développement des outils d'exploitation des ontologies qui sont un maillon important du Web sémantique (T. Berners-Lee, 2001). Parmi les outils basés sur les ontologies mis au point par IBM, nous trouvons *Integrated Ontology Development Toolkit (IODT)* : un outil dédié au stockage, à la manipulation, à l'interrogation et à l'inférence des ontologies et leurs instances. Dans le cadre de notre travail, nous nous sommes intéressés à SOR (Lu *et al.*, 2007) qui est un référentiel des ontologies OWL. SOR est incorporé dans IODT (Integrated Ontology Development Toolkit). Il utilise une architecture type II ("trois quarts") pour le stockage de l'ontologie et sa population. Il comprend un raisonneur et un outil de recherche. Il utilise pour le stockage une base des données relationnelles notamment DB2. Les outils de SOR sont destinés à manipuler les ontologies OWL. Mais, l'implantation actuelle manipule les ontologies OWL-Lite et OWL-DL.

Le Tableau 1 récapitule quelques caractéristiques essentielles des trois BDBO (Oracle, IBM SOR et Ontodb).

Caractéristiques	**Oracle**	**SOR**	**OntoDB**
Formalisme	RDF, OWL	OWL	PLIB
Échange de données	oui	oui	Oui
Support linguistique	oui	oui	Oui
Langage de requêtes	sql, sparl	sparql	ontoql, sparql
Extraction d'ontologies	Oui	oui	Oui
Sécurité des données	oui	oui	Oui
Gestion des versions et évolutions des ontologies	non	non	Oui
Cycle de vie des instances	non	non	Oui
Moteur d'inférence	oui	oui	Non
Base de règles utilisateur	oui	non	Non
Exportation des données	oui	oui	Oui
Vérification de la consistance	Oui	Oui	Oui
Modèle de stockage d'ontologies	Verticale	Spécifique	Spécifique
Modèle de stockage des instances	Verticale	Binaire	Horizontale
Architecture	Deux quarts	Trois quarts	quatre quarts
SGBD actuellement utilisés	Oracle	DB2, Derby	Postgres

Tableau 1. *tableau comparatif d'OntoDB, Oracle et SOR.*

5. Chargement d'OWL dans OntoDB

Bien qu'OntoDB soit prédisposé à prendre en compte tout formalisme d'ontologies ; il n'est actuellement équipé que d'un module de chargement d'ontologie PLIB.

Vu que l'ontologie que nous souhaitons utiliser pour nos tests a été représentée en OWL, nous devons proposer un moyen pour charger cette ontologie dans OntoDB. Une solution possible serait de faire un programme permettant d'obtenir une ontologie PLIB à partir d'une ontologie OWL et la charger dans OntoDB. Mais, vu la différence et la diversité de constructeurs et des opérateurs des deux formalismes, une ontologie obtenue de cette manière n'est ni congruente ni équivalente à l'ontologie de départ. Nous avons donc préféré développer un module de chargement d'OWL dans OntoDB.

Ainsi, nous avons mis en place un module d'importation des ontologies OWL et leurs instances dans OntoDB. Ce module prend une ontologie OWL et ses instances et, après une analyse, il extrait les concepts (classes et propriétés) et leurs instances, puis il crée de la base des données et la remplit de ses instances. Nous nous sommes intéressés à la partie *canonique* de l'ontologie OWL (les concepts primitifs).

De manière formelle, notre module est une application d'appariement des concepts du formalisme d'ontologie OWL en concepts PLIB disponibles dans OntoDB. Si on note T, cette application, $T :< O_{OWL}, E_{owl} > \rightarrow < O_{OntoDB}, E_{OntoDB} >$ où O_{OWL} et E_{OWL} sont respectivement une ontologie et un concept ou instance dans cette ontologie dans le formalisme OWL, O_{OntoDB} et E_{OntoDB} sont respectivement ontologie et un objet(table, attribut ou valeur) dans OntoDB (en formalisme d'ontologie PLIB). Soit C, P l'ensemble de classes et des propriétés canoniques de l'ontologie O en OWL. T se base sur deux considérations principales :
$\forall c \in C$, $T(O_{OWL}, c)$ est classe de PLIB alors elle est définie dans OntoDB, et une table correspondante est créée pour ses instances. $\forall p \in P, T(O_{OWL}, p)$ est propriété dans PLIB et est attachée à Domaine(p) i.e. toutes les classes qui sont domaines de la propriété p.
Il faut noter que cette correspondance n'est pas parfaite et elle est irréversible du fait que les concepts OWL non canoniques ne sont pas pris en compte et qu'il n'est pas possible de retrouver l'ontologie initiale par une application inverse. Dans la section suivante, nous allons utiliser notre module pour charger les ontologies OWL dans OntoDB et étudier les performances des trois systèmes ci-dessus présentés en termes de temps de chargement des données et de traitement de requêtes.

6. Etude de performances

Dans cette section, nous menons une batterie d'expérimentation pour évaluer les trois SGBD étudiés selon deux critères : le temps de chargement des instances ontologiques et le temps d'exécution de requêtes. Dans les sections suivantes, nous décrivons les ensembles de données sur lesquels les expérimentations sont effectuées et leurs interprétations.

6.1. *Ensemble de données pour les tests*

Pour disposer des données ontologiques nécessaires pour nos tests de performance, nous avons utilisé le banc d'essai LUMB utilisant le formalise OWL. Il crée une ontologie de domaine universitaire. Chaque université est constituée de 15 à 20 départements. Dans chaque département, nous retrouvons des *enseignants* de différentes *catégories*, *des étudiants*, *des cours*, etc. Nous avons généré trois ensembles d'ontologies ayant respectivement 1, 5 et 10 universités (Lumb01, Lumb05 et Lumb10). Le nombre d'instances ontologiques dans chaque ensemble, leurs triplets sont illustrés dans le tableau 2. Notons que les outils proposés par Oracle ne chargent que des données au format N-TRIPLE (sujet, prédicat, objet). En conséquence, une conversion des données de notre banc d'essai exprimées en OWL vers le format cible d'Oracle fut nécessaire. Pour ce faire, nous avons utilisé l'API Jena appelé rdfcat.
Nos évaluations ont été réalisées sur un ordinateur (DELL) doté d'un Processeur Intel(R) Xeon(R) CPU E31225 à 3.10 GHZ et d'une mémoire vive de 4GO et d'un disque dur de 500Go. Les systèmes de gestion bases de données (SGBD) utilisés sont : Oracle 11g pour la base de données Oracle, IBM DB2 9.7 pour SOR et PosgresSQL 8.2 pour OntoDB.

Dataset	Nombre d'instances	Nombre de triplets
Lumb01	82415	100.851
LumB05	516116	625.103
LumB10	1052895	1.273.108

Tableau 2. *Ensemble de données générées.*

6.2. *Performances en termes de chargement des données (ontologie et instances)*

Cette partie concerne les chargements de l'ontologie et ses instances dans nos BDBO. Notre mode opératoire se résume en quatre points : (1) la génération de données OWL ; (2) la conversion et concaténation de ces données dans un fichier N-Triple pour Oracle ; (3) le calcul du temps de chargement (en sec.) des instances dans chaque BDBO ; et (4) l'interprétation des résultats.

Pour avoir des résultats homogènes, nous avons effectué quatre fois le chargement de chaque ensemble de données et nous avons retenu la moyenne obtenue.

Pour chaque BDBO, nous avons chargé et mesuré le temps de chargement des nos ensembles de données. Les résultats sont consignés dans le tableau 3.

6.2.1. *Le Chargement de données pour Oracle*

Pour Oracle, nous avons fait le chargement global (*Bulk Load*) et le chargement par lot (*Batch Load*). Pour le premier, nous avons mesuré le temps de chargements des données dans la table de relais (appelée *Staging Table*) et le temps de transfert

dans le modèle sémantique. Pour réaliser cette évaluation, nous avons utilisé l'utilitaire d'Oracle Sqlloader qui renvoie le temps qu'il met pour charger les données. Pour le temps de transfert, nous avons utilisé le chronomètre d'oracle. Le temps de chargement est alors égal à la somme de ces deux temps.

Pour le chargement par lot, la mesure du temps a été facilitée par l'API utilisée. En effet, à la fin du chargement, un résume du déroulement du processus est renvoyé et le temps mis est affiché. Dans les deux cas, après chaque chargement, nous supprimons et recréons le modèle sémantique, avant d'effectuer un autre chargement.

6.2.2. *Le Chargement de données pour IBM SOR*

Pour faire une évaluation de performance de SOR, nous nous servis de l'IODT fourni (http ://www.alphaworks.ibm.com). Nous avons utilisé comme SGBD, IBM DB2 Express 9. Nous avons chargé les différentes ontologies LUMB générées dans SOR. A la différence de chargement sous Oracle, les données n'ont pas subi de transformation de format, elles sont restées en format OWL. Pour chaque ensemble de données, nous avons mesuré le temps mis pour son chargement.

6.2.3. *Le Chargement de données pour OntoDB*

Nous avons utilisé notre module (décrit précédemment) pour le chargement d'ontologie OWL d'OntoDB dans le même environnement que pour SOR. L'utilitaire d'OntoDB permettant de charger des ontologies, ne travaille que sur des ontologies *acycliques*. Cela est une exigence de modèle d'ontologie PLIB qui est au coeur du système OntoDB. Or nos ontologies LUMB renferment des cycles. Nous avons rompu les cycles en transformant les propriétés de type *objectproperty* à l'origine des cycles en *datatypeproperty*.

Dataset	Oracle Batch	Oracle Bulk	IBM SOR	OntoDB
Lumb01	42	12	90	15975
LumB05	222	55	590	72699
LumB10	302	116	1147	146023

Tableau 3. *Récapitulatif des temps de chargement (sec).*

6.2.4. *Interprétation des résultats*

En terme de chargement, comme on le constate sur le tableau 3, Oracle fournit des très bons résultats, suivi de SOR. OntoDB se montre lent. Pour Oracle, cela peut s'expliquer par le fait que peu d'opérations sont effectuées sur des données à l'insertion. Le SOR prend un temps nécessaire pour les traductions. La lenteur que l'on constate sur OntoDB s'explique par les multiples sous-requêtes présentes dans la plupart des requêtes d'insertion des instances. En effet, pour les champs référencés (les clés étrangères), OntoDB stocke les identifiants (oid). Lors du chargement, le système exécute des sous-requêtes pour chercher les *oid* des instances de ces champs. Par exemple, l'insertion d'une instance de GraduateStudent comporte au moins quatre

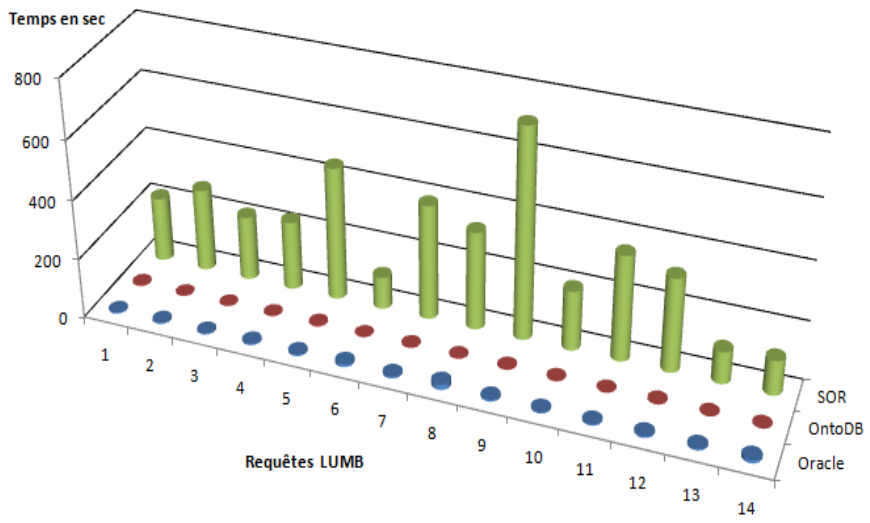

Figure 5. *Temps d'exécution de requêtes.*

sous-requêtes pour extraire les *oid* des cours suivis, du département d'appartenance, du *conseiller* (*advisor*) et de l'université d'où il était diplômé (undergraduateDegreeFrom). Pour éviter des erreurs lors d'insertion, OntoDB maintient un ordre qui impose la création de superclasses d'une classe avant celle de la classe et de même pour les instances. Tous ces mécanismes contribuent à une cohésion de données mais ne facilitent pas le passage à l'échelle.

6.3. *Performances en termes de temps de réponse de requêtes*

Nous avons mesuré les temps d'exécution des requêtes (en sec.) sur les trois ontologies. Comme pour le temps de chargement, pour chaque requête nous avons effectué quatre fois les tests et la moyenne est retenue. Les requêtes utilisées sont celles proposées dans le banc d'essai LUMB. Nous les avons traduites pour qu'elles soient acceptées par nos BDBO. Le jeu de données utilisé est Lumb01. Les résultats ont donné lieu aux histogrammes de la figure 5. Nous avons utilisé la base de règles *owlprime* pour Oracle.

6.3.1. *Interprétation des résultats*

Pour ce qui est de performance en termes de temps de réponses aux requêtes, OntoDB réagit mieux qu'Oracle et SOR. En effet, OntoDB, ayant stocké des *oid* des champs référencés, se sert de l'opérateur de référencement (*DEREF*) qui s'appuie sur ces *oid*, évitant ainsi certaines jointures. D'où le bon temps de réponse. Oracle vient en second lieu suivi de SOR qui est lent car il réalise plusieurs jointures pour toutes

les requêtes. Par contre, SOR réalise un bon travail de déduction. Par exemple à la requête 12 du LUMB, OntoDB et Oracle ne retournent pas de résultat, car il y a pas de déclaration explicite d'instance de *Chair* or SOR a réussi à renvoyer des résultats en se basant sur la propriété *headOf*, puisque un individu instance de *Professor* est instance de *Chair* s'il est à la tête d'un *Département*. Si on ajoute une règle traduisant cette assertion à Oracle, il fournira des réponses. Mais ce n'est pas une règle prédéfinie dans ce système.

7. Conclusion

Durant ces dernières années les ontologies ont connu une importante évolution dans plusieurs domaines et a donc nécessité un fort besoin en termes de stockage de ces ontologies. Ainsi, des universitaires et des industriels ont proposé des solutions de persistance s'appuyant sur des SGBD existants permettant la gestion et l'interrogation efficace des données tout en assurant une volumétrie importante des données. De ce fait, de nouveaux formalismes et de modèles de stockages dédiés aux instances et aux classes ontologiques et d'architectures ont émergé. Pour faciliter la compréhension de cette diversité, nous avons présenté un état de l'art sur les ontologies, les formalismes, les modèles de stockage et les architectures utilisées par les SGBD cibles. Une confrontation de trois types de SGBD supportant les ontologies dont deux provenant du monde industriel (Oracle et IBM SOR) et une du monde académique (LIAS-ENSMA) basée sur deux critères : *le temps de chargement des instances* et *le temps de réponse de requêtes*. Pour réaliser cette étude, nous étions confrontés à un problème d'hétérogénéité entre les formalismes utilisés dans les SGBD étudiés : OWL (pour Oracle et IBM SOR) et PLIB (pour OntoDB). Pour résoudre ce problème, nous avons proposé une solution de traduction d'une ontologie OWL vers une ontologie PLIB. Les résultats obtenus ont montré l'efficacité des SGBD commerciaux en termes de temps de chargement des données ontologiques tandis qu'OntoDB a permis de mettre en avant ses capacités pour le critère de traitement de requêtes.

Concernant le temps de chargement d'OntoDB, nous expliquons cela par le fait que le langage de requête OntoQL a été développé dans le cadre spécifique des ontologies basées sur le formalisme PLIB. Par conséquent, la réduction du chargement pourrait s'effectuer en faisant abstraction d'OntoQL.

8. Bibliographie

Alexaki S., Christophides V., Karvounarakis G., Plexousakis D., Tolle K., « The ICS-FORTH RDFSuite : Managing Voluminous RDF Description Bases », *Proceedings of the 2nd International Workshop on the Semantic Web*, p. 1-13, 2001.

Bechhofer S., van Harmelen F., Hendler J., Horrocks I., McGuinness D., Patel-Schneider P., Stein L., « OWL Web Ontology Language Reference », *W3C, http ://www.w3.org/TR/owl-ref/*, 2004.

Bellatreche L., Dung N. X., Pierra G., dehainsala H., « Contribution of Ontology-based Data Modeling to Automatic Integration of Electronic Catalogues within Engineering Databases », *Computers in Industry*, vol. 57, n° 8-9, p. 711-724, 2006.

Brickley D., Guha R., « RDF Vocabulary Description Language 1.0 : RDF Schema », *W3C, http ://www.w3.org/TR/rdf-schema/*, 2002.

Broekstra J., Kampman A., van Harmelen F., « Sesame : A Generic Architecture for Storing and Querying RDF and RDF Schema », *Proceedings of the 1st International Semantic Web Conference (ISWC'02)*, p. 54-68, 2002.

Bruijn J. D., Polleres A., Lara R., Fensel D., « OWL Flight », *WSML Delivrable D20.3 v0.1, http ://www.wsmo.org/TR/d20/d20.3/*, august, 2004.

Connolly D., van Harmelen F., Horrocks I., McGuinness D. L., Patel-Schneider P. F., Stein L. A., « DAML+OIL Reference Description », *W3C, http ://www.w3.org/TR/daml+oil-reference*, March, 2001.

Dehainsala H., Pierra G., Bellatreche L., « OntoDB : An Ontology-Based Database for Data Intensive Applications », *DASFAA'07*, p. 497-508, 2007.

Fankam C., OntoDB2 : un système flexible et efficient de Base de Données à Base Ontologique pour le Web sémantique et les données techniques, Ph.d. thesis, Poitiers University, 2009.

Fensel D., Decker S., Erdmann M., Studer R., « Ontobroker : How to make the WWW Intelligent », *In Proceedings of the 11th Banff Knowledge Acquisition for Knowledge-Based Systems, Workshop (KAW98)*, p. 9-7, 1998.

Gruber T. R., *Formal ontology in conceptual analysis and knowledge representation. Chapter : Towards principles for the design of ontologies used for knowledge sharing.*, Kluwer Academic Publishers., 1993.

ISO-13584-42, Industrial Automation Systems and Integration Parts LIBrary Part 42 : Description methodology : Methodology for Structuring Parts families, Technical report, ISO, 1998.

Jean S., Ait-Ameur Y., Pierra G., « Querying Ontology Based Database Using OntoQL (an Ontology Query Language) », *ODBASE'06*, p. 704-721, 2006.

Jean S., Pierra G., Ait-Ameur Y., *Domain Ontologies : a Database-Oriented Analysis*, vol. 1, Springer, p. 238-254, 2007.

Klyne G., Carroll J. J., « Resource Description Framework (RDF) : Concepts and Abstract Syntax », *W3C, http ://www.w3.org/TR/rdf-concepts/*, February, 2004.

Lu J., Ma L., Zhang L., Brunner J.-S., Wang C., Pan Y., Yu Y., « SOR : a practical system for ontology storage, reasoning and search », *VLDB'2007*, p. 1402-1405, 2007.

Murray C., « Oracle Database Semantic Technologies Developer's GuideI », *Oracle Corporation*, Nov, 2008.

Park M. J., Lee J. H., Lee C. H., Lin J., Serres O., Chung C. W., « An Efficient and Scalable Management of Ontology », *DASFAA'2007*, p. 975-980, 2007.

Pierra G., « Context Representation in Domain Ontologies and its Use for Semantic Integration of Data », *Journal Of Data Semantics (JoDS)*, vol. 10, p. 174-211, 2008.

T. Berners-Lee J H. O. L., « The Semantic Web », *Scientific American*, May, 2001.

Wu Z., Eadon G., Das S., Chong E. I., Kolovski V., Annamalai M., Srinivasan J., « Implementing an Inference Engine for RDFS/OWL Constructs and User-Defined Rules in Oracle », *ICDE'2008*, p. 1239-1248, April, 2008.