Explorations in the Use of Semantic Web Technologies for Product Information Management

Jean-Sébastien Brunner [1], Li Ma [1], Chen Wang [1], Lei Zhang[1],
Daniel C. Wolfson [2], Yue Pan [1], Kavitha Srinivas [3]
brunner@cn.ibm.com, malli@cn.ibm.com, chwang@cn.ibm.com, lzhangl@cn.ibm.com,
dwolfson@us.ibm.com, panyue@cn.ibm.com, ksrinivs@us.ibm.com

[1] IBM China Research Laboratory
ZhongGuanCun Software Park #19
Dong Beiwang road, ShangDi
Beijing 100094, China

[2] IBM Software Group
11400 Burnet Rd.
Austin, TX 78758-3415
USA

[3] IBM Watson Research Center
P.O.Box 704,
Yorktown Heights, NY 10598
USA

ABSTRACT
Master data refers to core business entities a company uses repeatedly across many business processes and systems (such as lists or hierarchies of customers, suppliers, accounts, products, or organizational units). Product information is the most important kind of master data and product information management (PIM) is becoming critical for modern enterprises because it provides a rich business context for various applications. Existing PIM systems are less flexible and scalable for on-demand business, as well as too weak to completely capture and use the semantics of master data. This paper explores how to use semantic web technologies to enhance a collaborative PIM system by simplifying modeling and representation while preserving enough dynamic flexibility. Furthermore, we build a semantic PIM system using one of the state-of-art ontology repositories and summarize the challenges we encountered based on our experimental results, especially on performance and scalability. We believe that our study and experiences are valuable for both semantic web community and master data management community.

Categories and Subject Descriptors
E.2 [**Data Storage Representations**]

H.3 [**Information Storage And Retrieval**]

General Terms
Performance, Design, Experimentation.

Keywords
Ontology, Semantic Web, Master Data Management, Product Information Management, Modeling.

1. INTRODUCTION
With increased market competition, modern companies have become seriously dependent on their information at hand, such as customer data and product information. If such information were to be complete and accurate, companies can make business decisions agilely and correctly. Typically, however, such information is incomplete and inconsistent across many systems. As an example, a single product can have different codes and descriptions in the various markets it is sold in, and a single customer can have different IDs in various systems. This situation mainly results from the lack of global standards (or insufficient application of standards), as well as the fact that data is captured many times, in many different systems. This causes recurrent data alignment issues which hinder smooth and effective Inter-Market Supply operations. Therefore, it is critically important to maintain a single group of core entities across many systems within an enterprise to improve business efficiency and customer satisfaction.

Core business entities a company uses repeatedly across many business processes and systems are called master data [24]. Master data refers to lists or hierarchies of customers, suppliers, accounts, products, or organizational units. Product information is the most important kind of master data and product information management is becoming critical for modern enterprises because it enables companies to centralize, manage and synchronize all product information with heterogeneous systems and trading partners [10]. Recently, well-known data management solution providers, such as IBM, Oracle and SAP, have released their master data management (MDM) solutions [15][27][29], especially on customer data integration (CDI) and product information management (PIM). But, there are some open technical challenges, as reported in [24][10], for instance, federation and identity management. The most critical challenge is that MDM solutions need to be built on an enterprise-wide master data model which provides a logical model for aggregating and reconciling the various data sources that comprise a master data record. This common master model should be flexible to deal with business changes and expressive enough to represent the semantics of master data. Based on this flexible and expressive data model, a scalable and high performance MDM hub should be developed.

In this paper, we explore the use of semantic web technologies for product information management and believe similar technologies can be used for customer data integration. We investigate the use of ontology for expressive PIM representation and modeling, and build a PIM prototype on top of one of the state-of-art ontology repositories. Moreover, we summarize the challenges we encountered based on our experimental results. We believe that our study and experiences

WWW 2007, May 8–12, 2007, Banff, Alberta, Canada.
ACM 978-1-59593-654-7/07/0005.

are valuable for both semantic web community and MDM community. The rest of this paper is organized as follows. Section 2 introduces semantic PIM, including ontology's value for PIM . Section 3 details ontology modeling for PIM data representation. Section 4 describes the architecture of our developed prototype system. Experimental results on a real customer data set are reported in Section 5. Section 6 presents discussions and future work. Section 7 concludes this paper.

2. ONTOLOGY'S VALUES FOR PRODUCT INFORMATION MANAGEMENT

A flexible and expressive enterprise-wide data model is the key to efficient Product Information Management. In this section, we interpret the value of ontologies for PIM representation.

An ontology is often defined as an explicit specification of shared conceptualization, and is a key structure for knowledge representation, especially to represent complex or changing data. Here we focus on standardized means of knowledge representation, such as recommendations developed by the World Wide Web Consortium: the Resource Description Framework (RDF)[22], the RDF Schema (RDFS) [3], and the Web Ontology Language (OWL) [31] which correspond to three nested ontology models. As the cornerstone of information representation and exchange, RDF defines a simple model extended by RDFS which introduces class support. OWL introduced some axioms based on description logic (DL) paradigm, giving the possibility of rich and flexible class definitions.

Product information is subject to continuous changes due to the introduction of new products, evolution of products, changes in the organization due to internal reorganization or fusion-acquisition (fusion of product lines, organization of hierarchies or policies). Such changes are often difficult to capture in traditional relational databases, and it is our conjecture that they can be more accurately captured by a declarative ontology.

The ontology's values presented in this section are illustrated on Figure 1, based on a real example of Electronic Consumer products defined using OWL constructs.

RDF uses the concept of Universal Resources Identifiers (URIs) as Web-based identification scheme, which make ontologies inter-operable and facilitate integration and communications of several core-ontologies. In practical terms, it leads to two distinct advantages: on the one hand, it allows one to refer to industry specific standards; and on the other hand it allows synchronization of product information management utilities to other core business entities, such as customer data integration (CDI).

RDFS defines the data in the form of a labeled edge graph, but also provides the ability to define **class inheritance**. It enables the definition of a hierarchical model that is easy to understand and close to both business objects and OO classes (such as in Java). RDF features are then predominant in PIM models using category trees. If a catalog defines a "Computer" category as a class, and then defines "Laptop" as its subclass, the "Laptop" will have automatically the properties "CPU speed" and "Memory" and can add new properties such as "Screen Size".

OWL allows the definition of **richer properties and relationships**. In OWL, relationships are divided into two types (*Object Properties* defining relationships between individuals and *Datatype Properties* defining a literal value). In OWL, it is possible to define an Object Property as symmetric, functional, inverse functional, or transitive. OWL Object Properties are then suitable to describe the complex relationships among products and between products and other entities in the product information. Followings are some examples of product relationships that require such complex relationships include: *packaging*, *substitution*, *cross-sell*, *up-sell*. Transitivity could ensure the exact semantics of the *substitution* relationship.

The expressivity of OWL allows the definition of **logical classes** (*intersection*, *union* and *complement* operators), which enables automatic classification for product items. OWL Restrictions support the creation of **dynamic categorization** of product lines, as shown in examples below.

In PIM, logical classes will often use intersection, such as the definition of new product categories which intersects two others. For examples, *smartphones* products, which gather characteristics of PDA and phones are a good example: any product which is simultaneously a PDA and a phone is then a *smartphone*.

Dynamic categories use OWL restrictions as basis and can represent complex and potentially evolving categories. Here are some examples:

- *Minimum cardinality restriction* can define an "outdated products" category which gathers all products replaced by at least one other product;
- *AllValue restriction* on "made by" property to "metal" class can define a "metallic products" category;
- *SomeValue restriction* on "composed of" to "battery" can define a "battery powered products" category;
- *HasValue restriction* on "made by" property to "aluminum" individual can define a "aluminum products" category.

Such restrictions are also used to define specific **product templates**. For example an "IBM LCD Display" can be described using the following restrictions: {color=black, manufacturer=IBM, power=110-240V, etc}. This avoids any unnecessary duplication in the data and simplifies the maintenance of data.

The flexibility, expressivity and ease of integration that ontologies provide make them excellent candidates for product information modeling.

3. ONTOLOGY MODELING FOR PIM

Based on the requirements of PIM expressed by real customers, we propose here a multi-layer model for product information representation through OWL ontology language. Since OWL is a MOF 2 compliant model [28], we can express OWL in term of this OMG recommendation [23]. This layered-model, illustrated on Figure 2 relies on MOF meta-meta model, its meta- level (M2) is based on OWL-DL ontology language, enriched for practical use by meta-modeling features and N-ary relationship specific support. Based on this meta-level, the other layers express definition of product information. Models using meta-modeling can be difficult to express in term of MOF layers, but it is possible to comprehend them using two kinds of instantiation as introduced in [2]. So, from Figure 2, it can be

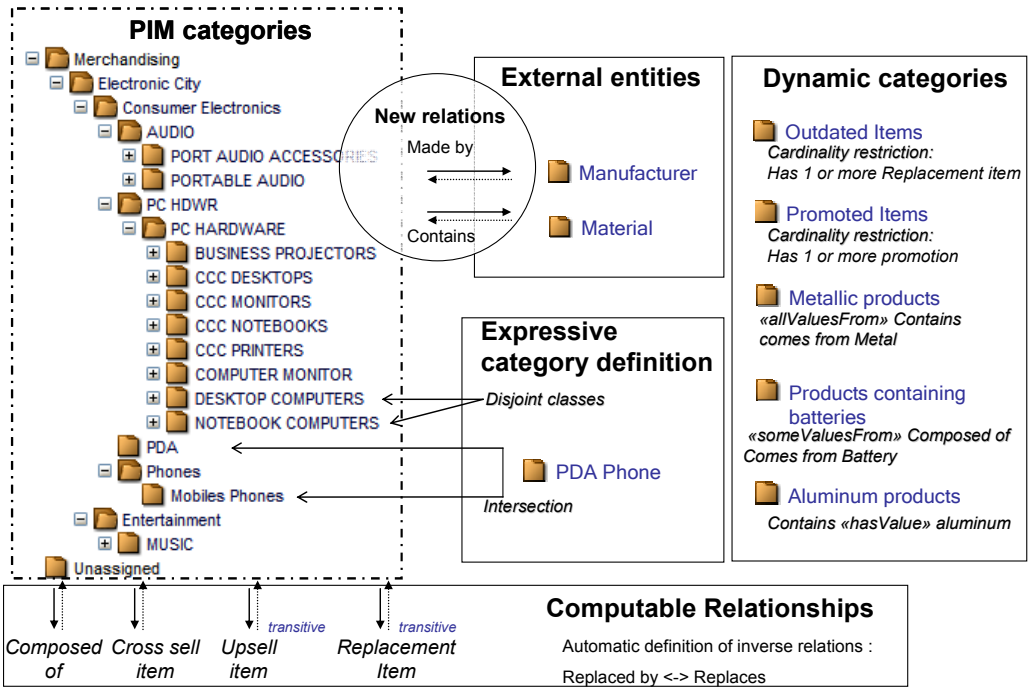

Figure 1. PIM modeling using OWL expressivity

observed that instantiation is used for two purposes. First, to cross the MOF layers and link objects to their classifier (labeled as meta-model instantiation on the figure), and secondly to instantiate some semantic objects (labeled as model instantiation) by using meta-modeling inside the M1 level.

These layers are described in more details in the following.

3.1 Meta Model Based on OWL

3.1.1 A Practical Choice of OWL Dialect

The OWL language became a widely used knowledge representation language due to its powerful expressivity and answers to many requirements expressed for PIM modeling, as argued in section 2. In fact it is divided into three sub-languages with growing expressivity and complexity of reasoning: *OWL Lite*, *OWL DL* and *OWL full*.

OWL Lite, offering the minimum expressivity of OWL, cannot support some constructs like *hasValue*, *disjointWith* which can express valuable information in the product information. At the opposite *OWL full* offers a comprehensive modeling possibilities but lead to computational problems for reasoning.

For PIM modeling and reasoning, we found that only a subset of OWL DL is needed. Practically the needs of reasoning can be constrained to a subset of DL for most of PIM requirements. A subset, known as *OWL DLP* (Description Logic Program) [12] represents a good trade-off regarding this context. As a consequence the reasoning capabilities will be mainly based on DLP. Nevertheless, more expressivity can be used in the modeling to define a complete and accurate model, as we can see in later sections

3.1.2 Meta Modeling

Within the different OWL dialects, only *OWL full* allows meta-modeling, but this language is intractable, resulting in limited use for meta-modeling in practical ontology models. Yet, in real business models, meta-modeling is often needed because the distinction between *classes* and *instances* is not always trivial to model the real world [30][32].

As an illustration in PIM, we can take a simple model defining the concept of *Laptop*, *Categories* and *Products*. The common way to structure this model is to define *Laptop* as an instance of *Categories*. In such a model, *Laptop* can be described by values (e.g. *contact information of this Category*, *whether it is possible to sell this category online*, *accounting information*, etc.). But using this model, it is not possible to define a particular product, such as "My Laptop", as instance of *Laptop* category, because this one is already defined as instance. So in many models, categories are just defined as classes with limited capabilities to avoid recourse to meta-modeling.

In our model, we chose to allow this meta-modeling, notably to support rich *Category* description and also to give more details on *Properties*.

Additionally, there is a limitation in the MOF2 semantic which makes impossible to have an object instantiating two classifiers at the same time. As a consequence, some simple workarounds are needed to allow OWL full meta-modeling style [28]. As represented on Figure 2 it consists in adding some association classes at M2 level.

The computational problem raised by the introduction of meta-modeling can be solved by the use of *punning* semantics, which insures that the different aspects of entities (Individual, OWL Class) are considered separately, as described in section 6.1.1.

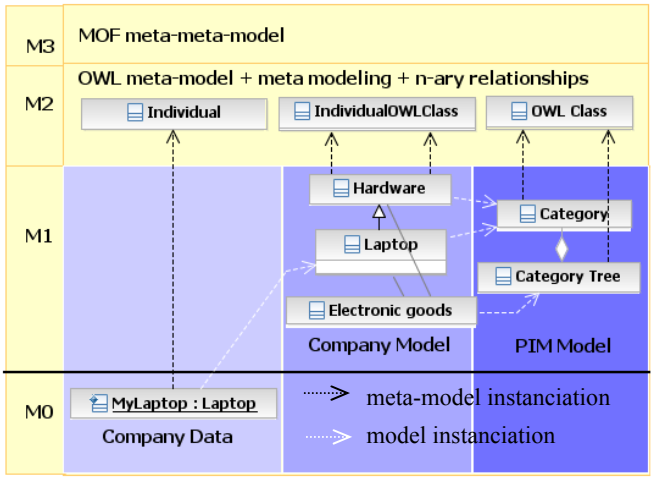

Figure 2. Model stack for Product Information Management, in comparison to MOF levels.

3.1.3 Located Properties

In product information management, some N-ary relationships are often used to describe the variability of information. Properties can be time-dependant, such as a price which can vary periodically. Location is another important feature in PIM since many properties, such as price, or shipping can be location-dependant. Location dependant properties exist also to support the local legislations or regional specificities in a global product information management solution. For instance, the list of components of a product or its manufacturer can evolve from one country to others. At a larger scale, the prices can vary not only according to the countries or the regions but also according to the specific retailers. Taking the example of the retailing giant Wal-Mart: up to 142,000 different products can be sold in one of their 6500 stores. In this context nearly one billion prices could be stored and retrieved, but in order to avoid redundant entries, some prices can be deducted from the hierarchical location structure. Indeed price policy is often structured by countries, regions and areas. Due to tree structure of these locations, the price can then be *inherited* from the parent location, or overridden in the case the retailer has a non-standard price policy. This involves the use of hierarchy-based conditions in the n-ary relationships.

Some other knowledge representation paradigms, such as F-logic[17], a frame-based language, allow the definition of complex and non-monotonic inheritance more naturally. However, since is not a W3C standard, we did not focus on this paradigm, to insure a better interoperability with other systems.

Regarding OWL, it defines only binary relationships but allows modeling any N-ary one using several techniques [13]. In this case we can use a tree to model the locations while property values and locations are stored in an intermediate class. Therefore modeling such relations is not a problem, but the implementation of located property behavior is out of scope of OWL and needed to be addressed by the application layer. To improve this point, we added a native support to store and retrieve efficiently the values of these properties. This is described in section 6.1.2.

3.1.4 Class Variables

In Oriented-Object modeling, programmers have the possibility to use *class variables*, also known as *static variables* in some languages. These variables have the special feature to have their value shared by all instances of the same class. For instance, if you define a given type of bank account, you can set directly the *interest rate* of all its instances by changing this unique property value.

In OWL, this could be done using a "hasValue" restriction to the property "interest rate" for the class "Saving Account". As a consequence, the reasoning will propagate this value to all the instances of this account. This kind of value propagation avoids users to maintain unnecessary data and allows instantaneous query on the characteristics of the products.

Whereas this feature is defined in OWL DL modeling capabilities, it is out of scope of DLP semantics used for reasoning. In consequence it required a specific support in the reasoning engine.

3.2 Product Information Management Model

Since every industry and customer has their own specific model, the intrinsic definition of products and their relationships are very different and prevent the development of a generic data-model. Based on our experience with representing products for many customers, we therefore defined a simple but effective meta-model to allow companies define their own model.

The Product Information Management model is an ontology representing the core elements of PIM data models. This ontology, mainly composed of *meta-classes*, is expressed in terms of our meta-model based on OWL and so belongs to M1 level and needs to be customized by a company model in order to be used. We present hereafter only several entities of this model.

Catalog: A meta-class defining the concept of catalog; the instantiation at the company model level of the catalog meta-class will also be a class. An instance of this meta class is structured by one or several hierarchies, and will define the properties applied to all products it contains.
Category Tree: A meta-class for category trees; its instances are used to give a structure to the catalogs instances and then to classify products.
Instances of these meta-class hierarchies will be the category trees used in the company model.

Category: A meta-class for categories. This meta-class allows defining categories with some specific properties, such as the *"Laptop"* category given as example. These instances will have some valued properties describing them in more details and, at the same time, will be used to classify products and define new properties for these products. A particular product "My Laptop" will be consequently an instance of an instance of this meta-class, as illustrated on the Figure 2.

Categories are then structured thanks to the *specialization (subclass Of)* relation. This specialization is compatible with the concept of categories which are defined from the most general one to the most specific ones.

Organization-Location Tree: This meta-class is conceptually very similar to the *Category Tree* meta-class since it is used to

define hierarchies. But here, *Location* or *Organization nodes* replace *Categories*.

Organization or Location nodes: These nodes are used to create the nodes of the *Organization-LocationTree*. In this case, the exact semantics of the tree is different since the tree is structured by a "part of" relation: a location is a part of the parent location. But conceptually, the subclass relation can also be used since we may need to define some attributes at a given level and, thanks to the inheritance tree, the children will have the same attributes.

PIM model also defines some regular OWL Class providing common PIM entities needed to describe products, such as Packaging, Suppliers, Manufacturers, etc.

As an instance of this model, the company model will be insured to follow this PIM model and be compliant with tools included in our PIM framework.

3.3 Company Model

The company model is the data model defined by the company to model its own business; it defines notably instances of catalogs, category trees and their categories, as well as properties of all these objects.

Defined in OWL as instance of the provided PIM model, this model bears the semantics of PIM objects, while being highly flexible. This model can be edited by a generic ontology-editor supporting meta-modeling or by a workbench we developed specifically to edit OWL file on top of our PIM meta-model.

Finally the data, at the M0 level, will use all classes available in upper layers. Data will instantiate classes from the company model, and also some classes directly defined in the PIM model. Products instances, one of the most important data in PIM, will instantiate simultaneously the instance of *Catalog* and the instance(s) of *Categories* they belong to; as a consequence they will benefit from their respective properties.

4. ARCHITECTURE

In this section we present the architecture of our prototype system on top of an ontology repository. This prototype is structured around two main components, as illustrated in Figure 3. One is a modeling workbench, and the other is an ontology repository.

This architecture was designed to be simple but inter-operable. It is notably built on OWL and SPARQL, making the data easily accessible by a wide-range of programs.

4.1 Product Information Ontology Workbench

Various ontology editors were developed to create and edit OWL ontology, such as Protégé 2000 [19] and SWOOP [16].

Although they are helpful in ontology development, we observed that they are not easy to comprehend by business users, who are more used to modeling languages such as UML. Additionally, OWL editors such as Protégé or SWOOP are inadequate for meta-modeling, especially, for classes which are at the same time defined as *OWL individuals*, as described in the previous section. Because oriented-object modeling (such as UML) is widely accepted and understood in the business world, this workbench uses some similar way to define model using OWL. But in such case, meta-modeling is not easy to represent when designing a model.

There is some interesting work [20] about the use of UML to clearly represent the meta-modeling layers, and to define the depth to which a model element can be instantiated. This approach also elegantly defines which properties are applied to instances and which properties are applied to classes. Such a representation can make the modeling task clear and accurate and allows the correct design of user interfaces dealing with OWL meta-classes. As a result, properties can be clearly defined as properties describing the class, or properties describing the instances, and eventually as properties describing the instances of these instances.

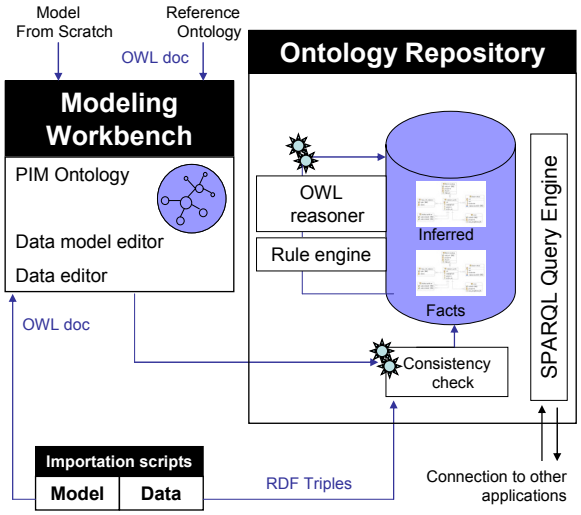

Figure 3. Semantic PIM architecture

We developed a Product Information Ontology Workbench based on these requirements for modeling and meta-modeling. This workbench is able to load and save OWL (from file or from our repository), and it uses the PIM meta-model we developed (described in section 3) to create company defined models. Most of the complexity of OWL is hidden in order to highlight the business concepts: hierarchies, categories of products, locations, relationships, etc. Additionally, the meta-modeling and DL-specific constructs are defined in an intuitive way according to our meta-model. For instance, for a *IndividualOWLClass*, we can view and edit simultaneously both its class and instance aspects. OWL restrictions are also created by specific wizards when needed.

4.2 Ontology Repository

In the particular context of product information management, models often include many different categories of products organized in taxonomies. Typically, different products will define very different properties. For example a computer will be notably described by "CPU speed" and "Memory" whereas a camera will be described by "Number of pixels" and "Zoom". A catalog can thus define several thousands of properties. A persistent store based on a "horizontal table", i.e., a table that

defines as many columns as properties in the model, cannot be used for two main reasons. The first one is that relational databases have a limitation to the number of possible columns in a table (generally 1012). The second reason is that among all columns, only a small percentage will be valued for one specific type of category, making this table very sparse. Using a different table for every product category is also possible but leads to a lack of flexibility since a modification to the product catalog will need an update to the database schema. This problem, shared by both e-commerce and product information management has been already studied [1] and was solved by the use of a generic *vertical table* that stores all possible property values (as illustrated on Table 1). As a result, such a schema stores some triples (product, property, value), similar to RDF triples: (S,P,O) for Subject, Predicate (also called property), Object (target of the relationship, or value). This storage style is implemented in commercial products such as Websphere Commerce or Websphere Product Center, but these products usually does not use *open standard* like OWL, which limits the expressivity and evolution of the model.

In the past years, several ontology repositories have been developed, either based on traditional relational databases management systems (RDBMS) such as Sesame [4] or Jena [6]; or repositories that have developed their own ontology schema (e.g. native ontology store, OWLIM [18], HSTAR [7]). The first family – also known as triple stores - leverages the benefit of RDBMS and benefits from additional features such as concurrency and transactional support, while the second kind adopts a native schema, closer to ontology and performs faster operations on ontology objects.

Coincidently, most PIM repositories and ontology triple stores use vertical schema in RDBMS.

Table 1 illustrates the vertical storage model where the principle is to use a main table storing the RDF triples on the form (S,P,O). Such a table can contain all the properties of all products and does not need any modification of the table structure in case a new property is needed.

Table 1. Example of vertical table

Subject	Predicate	Object
Product1	Reference	1001
Product1	Price	490
Product1	DisplayType	VGA
Product2	Reference	1002
Product2	Price	365

We based our Semantic PIM repository on a new version of Minerva [33] an ontology repository previously developed by our team. This new version, named SOR (Scalable Ontology Repository), is enhanced for PIM. Contrary to other RDBMS approaches which persist OWL ontologies as a set of RDF triples and do not consider specific process for complex class descriptions generated by class constructors (Boolean combinations, various kinds of restrictions, etc), our repository takes into account the specificities of OWL ontology and PIM requirements from its conception.

The first reason is to improve the reasoning on large number of instances managed in PIM. TBox reasoning, that is to say the reasoning on terminology box including classes and properties, is implemented by a DL reasoner (Pellet and Racer are currently supported) which can provide complete and sound DL reasoning. Considering the limited scalability of a DL reasoner on ABox (Assertion Box, including assertions about data) reasoning, SOR translates OWL semantics into a set of rules which can be easily implemented using SQL statements on RDBMS, thus supporting DLP expressivity in ontologies. This greatly improves the scalability of SOR. The benefit of our database schema is that all predicates in the body and head of the ABox inference rules have their separate tables in the database. Therefore, these rules can be easily translated into sequences of relational algebra operations. For example, the rule *Type(x,C) :- Rel(x,R,y), Type(y,D), SomeValuesFrom(C,R,D)* [1] has four predicates in the head and body using three different types; and so results in three tables: Relationship, TypeOf and SomeValuesFrom. Then a rule can be implemented using SQL *select* and *join* operations among these three tables. This effective integration of ontology inference and storage using DL-reasoner for TBox and DLP expressivity for ABox reasoning is expected to significantly reduce inference costs. SPARQL queries are supported to query ontologies in SOR. SPARQL queries are firstly translated into a single SQL statement which is evaluated on both explicit assertions and inferred results materialized in the persistent store, benefiting of decades of DB optimization.

SOR is used to store both model and data, to perform consistency check (e.g. see whether some DB-style constraints are infringed, as described in [26]), perform reasoning and allow queries on top of this data model.

5. EXPERIMENTAL RESULTS

Beyond expressivity, the other main point for PIM ontology storage was to scale it to millions of products (hundreds of millions of RDF triples).

In order to scale-up the ontology repository for PIM, we performed several experiments to measure the scalability of our ontology repository, Minerva, which thus far has the best performance on the extended ontology benchmark [21].

The experiments conducted here used real customer data, containing 4.2 millions of product items, which is about 120 millions of triples (4M data set). The product information we used contains an *Electronic Part* catalog and was converted to ontology format and stored in our ontology repository. The characteristics of products in this catalog are the following:

- 53 data type properties;
- 4 object properties ("parentPart", "hasManufacturer", "hasSupplier", "editedBy"), linked to classes *Product*, *Manufacturer*, *Supplier* and *Editor* respectively;
- Products are organized into 20 categories.

This model is pretty simple compared to the PIM expressivity previously described, but enabled to measure performances on this core product information.

[1] *SomeValuesFrom(C,R,D)* signifies that a particular class *C* may have a restriction on a property *R* that at least one value for that property is of a certain type *D*.

The experimental environment is a *blade server* with an Intel Xeon 2.8 GHz CPU and 4 GB RAM.

Table 2. Representative queries used in the experiments

#	*Query* *SPARQL*
Queries on properties	
Q1	*Retrieve one product according to its ID* SELECT * WHERE {?x pim:PartKey '34389'}
Q2	*Retrieve all products made by a given supplier* SELECT ?x WHERE {?x pim:hasSupplier pim:SUP1}
Queries involving relationships	
Q3	*Retrieve all products whose parent is made by a given supplier* SELECT * WHERE {?x pim:hasParent ?y . ?y pim:hasSupplier pim:SUP2}
Q4	*Q3 extended with a filter* SELECT * WHERE {?x pim:hasParent ?y . ?y pim:hasSupplier pim:SUP3 . ?x pim:SCode ?b FILTER (?b != 'N')}
Queries where predicate (property) is a variable	
Q5	*Retrieve all information about a given product* SELECT * WHERE {?x ?y ?z . ?x pim:PartKey '34389'}
Q6	*Retrieve all products and relationships to a given company* SELECT * WHERE {?x ?y pim:ABC}
Queries on the class structure (intersections)	
Q7	*Retrieve all products belonging at the same time to category C1 and C2* SELECT ?x {?x a pim:C1 . ?x a pim:C2}
Q8	*Class intersection and manufacturer != supplier* SELECT ?x {?x a pim:C3 . ?x a pim:C4 . ?x pim:hasSupplier ?y . ?x pim:hasManufacturer ?z FILTER (?y != ?z)}

For the experiments, we used the queries listed in Table 2. They represent a representative set of query patterns that could be performed on the given catalog. Other queries using the same patterns but different constraint values were used to confirm these results.

The query time, increasing in polynomial time according to the size of data set, could reach previously hundreds of seconds, and even more, in Minerva as shown on Table 3.

From these results we can see that Minerva had some difficulties on such a large dataset, especially when the number of joins involved in the query increased. In order to improve these results and be able to use Semantic Technologies for storing product information management, we are developing a new version of Minerva, called SOR, which contains some substantial changes in the schema. This version is still in development but we wanted to highlight the huge performance gains we have currently obtained and measured on this large scale data set, queried by some customer representative queries. Optimization schemes used in SOR include: creating Multiple Dimension tables to reduce IO costs, separating object relations with data type attributes in the physical schema, optimized index for representative queries, and the use of efficient hash codes to improve string search. The right column of Table 3 demonstrates the overall benefic of these improvements, with several typical query times below 1 second on the full data-set. Nevertheless, even if all query runtimes were improved, we can see performance on some queries patterns (involving a variable predicate or class intersections) still need to be improved.

Table 3. Comparison on ontology storage before and after improvements (4M data set)

Query	Result size	Runtime (ms) Minerva	SOR
Q1	1	**245,812**	**146**
Q2	137	**162,172**	**21**
Q3	6,181	**245,594**	**646**
Q4	1,341	**19,430**	**484**
Q5	33	**3,233,633**	**16**
Q6	386,038	**364,586**	**18.088**
Q7	19	**18,375**	**4,709**
Q8	25	**330,845**	**16,755**

6. REMAINING CHALLENGES AND FUTURE WORK

Using the model described in the section 3, with meta-modeling and expressive OWL axioms, it is difficult to use any current top-of-the-box ontology repository which are limited in expressivity or and in scalability. That is why we investigated how to improve our previous ontology repository to efficiently store and retrieve PIM data. Regarding reasoning capabilities, even if meta-modeling is not used in the reasoning, the large scale of data involved in PIM also needed some specific consideration described in this section.

6.1 New features

6.1.1 Meta-Modeling.

Compared to current ontology repositories, which usually support some subset of OWL DL expressivity, we argued in section 3 that there is a need to develop a support of meta-modeling to store product information. The needed semantics for meta-modeling can be *Contextual semantics* [25] which corresponds to a new OWL proposal known as OWL 1.1 [8][14]. In practical terms, in this meta-modeling – also called *punning* – the same URI can be used simultaneously for an individual, a class, or a property while entities remain intrinsically different. Thus, no aspect of the use of the URI as an individual has any effect on the meaning of the URI as a class. As proved in [8], this absence of interaction between the

polymorphic forms is required by classical first order (description) logic fragments of OWL-Lite or OWL-DL and then allows reasoning.

It has to be observed that this semantics has some limitations, in particular in its application with rules. Actually rules applied to a class do not affect their instance interpretation. Taking an example with the following simple knowledge base:

ElectronicProduct(*Software*)
ElectronicProduct (C) $\wedge$ C(*I*) $\rightarrow$ *SoldOnLine*(*I*)

In the second assertion, C is firstly interpreted as an Individual, and then as a Class. In term of punning semantics, they are considered as two different objects, which make more difficult the combination between a generic rule language, such as SWRL, OWL[25].

As a conclusion contextual semantics gives the possibilities to enrich models without the need of complete re-implementation of reasoners, but in the meantime the extension to business rules will be more difficult.

For now reasoning on meta-modeling aspects is not implemented and rule support is not a priority in our product information model; allowing to use this semantics easily in SOR.

6.1.2 N-ary Relationship.

As N-ary relationships are often used in PIM, they need a specific support to improve storage and search efficiency. The case of located-in property described in section 3.1.3, which can need the definition of one billion values, pushed the need of specific storage. In order to avoid duplication of entities and so letting user only maintain the valuable information contained in the model, the storage takes full advantage of the tree structure of locations, combining hierarchy information to retrieve the information at the run-time.

This process requires an extension to the repository schema, and we are currently investigating a novel schema for storage and retrieval of N-ary relationships. It will use an extension of SPARQL supporting this specific storage and the propagation of values along hierarchies. This is some examples of the queries this repository can support:

To retrieve prices defined in a 3-ary relatationship (with location tree):

```
SELECT * WHERE {[hasLocation USA] ?product
hasPrice ?price}
```

To retrieve prices defined in a 4-ary relatationship (with location tree and timestamp):

```
SELECT * WHERE {[hasLocation USA]
[inCatalog fall2005]?product hasPrice
?price}
```

Note that in the current implementation; only one of the conditions of the n-ary relationship could have a tree structure.

6.1.3 Support of Dynamic Queries.

One significant advantage of using OWL restrictions for product category definition is that it allows the automatic classification of product items using ontology reasoning. Very often, users want to retrieve products satisfying a certain set of conditions in query time (such as, “find all batter-powered items”). But unfortunately, there is no such a category in predefined product ontologies. So, users can define a new category represented by an OWL restriction on the fly -such as, *battery-powered items* are defined as someValuesFrom(composedOf, Battery)- and make use of ontology reasoning to classify products automatically. This feature, showing the dynamic capability of semantic PIM representation, is highly attractive to business users. But, implementing runtime ontology inference on a large scale of master data is not trivial, and is still a critical challenge. Currently, we integrate IBM SHER reasoning engine [11] into our prototype system for dynamic queries. The preliminary results are promising and encouraging. We will continue to work on scalable ontology inference for dynamic queries.

6.1.4 Utility Functions

Utility Functions, as well as versioning of the ontology and its data or fine-grain access control, represent an important issue to meet business requirements.

These features may be crucial for adoption of semantic web techniques for product information management. Although transaction and concurrency is currently widely supported due to the underlying DBMS, support of access control and versioning requires specific efforts to deal with OWL data. Integration of these functions will affect storage, query and reasoning systems implemented in the repository, and hence is a key issue to explore.

6.2 Scale Up the Repository

6.2.1 Query Runtime.

While we demonstrated the value of ontology modeling for product information management, our performance is still unacceptable for very large data sets.

One major issue is the large size of the vertical table containing the triples, which in PIM case can reach more than hundreds of millions of records. This becomes a major issue in case of complex queries, even using several indexes. Several techniques such as the one described in [1] are currently being implemented to scale up the vertical storage and we already have promising results, but we are still working on a release of a highly scalable ontology repository, supporting large-sized ontology with millions of products.

6.2.2 Loading Time.

Another open challenge is the improvement of the loading time, especially for mass imports. The measured loading time in the current repository is at most 1000 triples by second; at this speed it takes several days to load the whole data set.

Preprocessing of customer data, optimization of ontology storage and fine tuning of RDMS will be an important focus for us to solve this issue.

7. CONCLUSIONS

This work shows the values of using ontologies and more specifically OWL to efficiently model product information. Use of ontologies allows capturing semantic relations between business objects. Business users can define their own model, customized for their particular business, using the defined meta-model.

Moreover, it appears that such practical utilization of semantic web technologies exceeds the expressivity and reasoning

capabilities of most of current ontology solutions, and in consequences, limits the current modeling capabilities. From this assessment we developed a new architecture integrating notably a specific editor, repository and reasoner enhanced for PIM model and data.

This new architecture demonstrated its added-value compared to existing solution, In particular the work on our new ontology repository, SOR, gave some very encouraging results and shows that performances can be significantly improved in order to use semantic web technologies for large data sets. However, our experiments also pointed that we still need to focus efforts to scale up ontology repositories before they can be used for PIM and MDM at a very large scale and allow runtime reasoning, notably for automatic classification. Scaling up repositories will allow the use of Semantic Web technologies in high-scale applications and will promote their use in product information management and other business core data.

8. REFERENCES

[1] Agrawal, R., Somani, A. and Xu, Y. Storage and Querying of E-Commerce Data, VLDB, 2001.

[2] Atkison, C. and Kuhne, T. Model-Driven Devlopment: A Metamodeling Foundation, IEEE Software, Sept/Oct 2003.

[3] Brickley, D. and Guha, R.V. RDF vocabulary description language 1.0: RDF schema. W3C recommendation, Feb 2004.

[4] Broekstra, J., Kampman, A., and Van Harmelen, F. Sesame: A generic architecture for storing and querying RDF and RDF schema. ISWC, 2002.

[5] de Bruijn, J., Franconi, E. and Tessaris, S. Logical Reconstruction of normative RDF. Proc. of the Workshosp on OWL Experiences and Directions (OWLED 2005).

[6] Carroll, J.J., Dickinson, I., Dollin, C., Reynolds, D., Seaborne, A. and Wilkinson, K. Jena: implementing the semantic web recommendations. WWW 2004.

[7] Chen Y., Ou J., Jiang Y., and Meng X. HStar - a semantic repository for large scale OWL documents. ASWC 2006

[8] Cuenca-Grau, B. OWL 1.1 Web Ontology Language Model-Theoretic Semantics, Editor Draft

[9] Customer Data Integration: Market Review & Forecast for 2005-2006, A CDI Institute MarketPulse™ In-Depth Report.

[10] Gartner Reports, Magic Quadrant for Product Information Management, 2006.

[11] Fokoue, A., Kershenbaum, A., Ma, L., Schonberg, E. and Srinivas K. The summary ABox: Cutting Ontologies Down to Size. ISWC 2006, 343-356.

[12] Grosof, B., Horrocks, I., Voltz, R. and Decker, S. Description Logic Programs: Combining Logic Programs with Description Logic. WWW, 2003.

[13] Hayes, P. and Welty, C. Defining N-ary Relations on the Semantic Web, W3C Working Group Note 12 April 2006

[14] Horrocks, I., Kutz, O., and Sattler, U. The Even More Irresistible SROIQ, KR 2006, 57-67

[15] IBM Websphere Product Center, http://www-306.ibm.com/software/integration/wpc/, 2004

[16] Kalyanpur, A., Parsia, B., Sirin, E., Cuenca-Grau, B. and Hendler J. Swoop - a web ontology editing browser, Journal of Web Semantics 4 (1), 2005.

[17] Kifer, M., Lausen, G. and Wu, J. Logical Foundations of Object-Oriented and Frame-Based Languages. Journal of the ACM, 1995.

[18] Kiryakov, A., Ognyanov, D., and Manov, D. OWLIM – a pragmatic semantic repository for OWL. WISE Workshops, volume 3807 of Lecture Notes in Computer Science, 182–192. Springer, 2005

[19] Knublauch, H., Fergerson, R.W., Noy, N.F. and Musen, M.A. The Protégé OWL Plugin: An Open Development Environment for Semantic Web Applications. ISWC, 2004.

[20] Kuhne, T. and Atkinson, C. The essence of multilevel metamodelling. In UML 2001, Number 2185 in Lecture Notes in Computer Science, 19-33. Springer, 2001.

[21] Ma, L., Yang, Y., Qiu, Z., Xie, G., Pan, Y. and Liu, S. Towards a complete OWL ontology benchmark. ESWC, 125–139, 2006.

[22] Manola, F. and Miller, E. *RDF primer*. W3C recommendation, Feb 2004.

[23] Meta Object Facility (MOF) 2.0, OMG Document: formal/2006-01-01, http://www.omg.org/cgi-bin/doc?formal/2006-01-01

[24] Morris, H.D. and Vesset, D. Managing Master Data for Business Performance Management: The Issues and Hyperion' s Solution, IDC white paper, 2005.

[25] Motik, B. On the properties of meta-modeling in OWL, ISWC 2005

[26] Motik, B., Horrocks, I. and Sattler, U. Integrating Description Logics and Relational Databases, Technical Report, University of Manchester, UK, 2006.

[27] Oracle Data Hub, http://www.oracle.com/data_hub/index.html, 2005.

[28] Ontology Definition Metamodel (ODM) Request for Proposal, OMG Document: ad/2003-03-40, http://www.omg.org/cgi-bin/doc?ad/06-05-01.pdf , 2006.

[29] SAP Master Data Management, https://www.sdn.sap.com/irj/sdn/developerareas/mdm, 2005

[30] Schreiber, G. The Web is not well-formed. IEEE Intelligent Systems, 17(2):79-80, 2002.

[31] Smith M.K., Welty C. and McGuinness D.L. OWL web ontology language guide. W3C recommendation, Feb 2004

[32] Welty, C. and Ferrucci, D. *What's in an Instance?* Technical report, Rochester Polytechnic Institute Computer Science Dept., 1994.

[33] Zhou, J., Ma, L., Liu, Q., Zhang, L., Yu, Y. and Pan Y. Minerva: A scalable OWL ontology storage and inference system. In Proc. of ASWC, 429-443, 2006.