SMDM: Enhancing Enterprise-Wide Master Data Management Using Semantic Web Technologies

Xiaoyuan Wang[1] Xingzhi Sun[1] Feng Cao[1] Li Ma[1] Nick Kanellos[2] Kang Zhang[3] Yue Pan[1] Yong Yu[3]
[1]IBM China Research Lab, China {wangxyxy, sunxingz, caofeng, malli, panyue}@cn.ibm.com
[2]IBM Software Group, Canada kanellos@ca.ibm.com
[3]Shanghai Jiao Tong University, China jobo@apex.sjtu.edu.cn yyu@cs.sjtu.edu.cn

ABSTRACT

Motivated by evolving business requirements and novel enterprise applications, we propose and implement the Semantic Master Data Management (SMDM), a semantics-level enhancement to the existing MDM solutions. The SMDM system publishes relational-based master data as virtual RDF store, and injects instantaneous reasoning capabilities into semantic queries. Two kinds of ontologies are introduced to the system, the core MDM ontology and the external imported domain ontology. SMDM enables data linking among multi-domains, implicit relationship discovery, and declarative definition and extension of business policies and entities. Based on these functions, modern companies can customize their applications and services on demand within the MDM hub. In the demonstration, we build the system environment based on IBM's MDM solution, and run the use cases on the master data of an insurance company.

1. INTRODUCTION

With increasing market competition and complex business performance management, it is critically important for modern companies to maintain a single group of core entities across many systems within an enterprise to improve business efficiency and customer satisfaction. There arises high demand for master data [16], which refers to core business entities a company uses repeatedly across many business processes and systems, such as lists or hierarchies of customers, suppliers, accounts, or organizational units. In recent years, well-known data management solution providers, such as IBM, Oracle and SAP, have released their master data management (MDM) solutions [4, 6, 8], especially on customer data integration (CDI) and product information management (PIM).

Driven by evolving business requirements and novel applications, the simple management of master data is far from satisfactory. A majority of modern companies need the semantic analysis and extension for core business entities and relationships, multi-domain collaboration and product reorganization, and valuable but implicit knowledge discovery in rich relationships. All of these pose great challenges to master data management. Motivated by this, we propose and develop the Semantic Master Data Management (SMDM), a semantics-level enhancement to the existing MDM solutions. To fully utilize the formal expressivity capability, we apply Semantic Web [9] technologies to the implementation of SMDM. The value of Semantic Web technologies has been witnessed by the emergence of open data applications, such as Wikipedia, DBpedia [2], and Freebase [12].

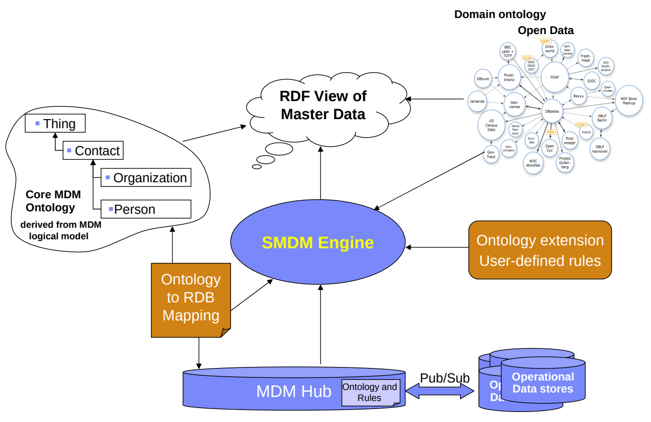


Figure 1: An overview of SMDM

Figure 1. illustrates the overview of SMDM. The SMDM engine publishes the underlying relational MDM database as virtual RDF store, which can link open data with master data to support effective collaboration. Two types of ontologies are introduced to the whole environment, the core MDM ontology and the external domain ontology. The former is derived from the MDM logical model containing core business entities and relationships, and the latter corresponds to external domains to be linked. To support SPARQL [11] (a query language for RDF data recommended by W3C), the SMDM engine translates a SPARQL query into a SQL statement, which will further be executed by the MDM relational database engine.

The technical challenges are how to effectively support query and analysis, and how to generate efficient SQLs for the underlying MDM database. Some previous work on exposing relational data as RDF data, such as D2RQ [1] and Virtuoso [5], lack the capability of reasoning for analysis, and pay little attention to the "quality" of the generated SQLs. Two aspects of technical highlights in SMDM should be mentioned. First, instantaneous rule reasoning is injected into query evaluation. With it, SMDM enables the capability of hidden relationship discovery and enriches semantic queries through the rule-based extension of SPARQL. Second, SMDM embeds a novel SPARQL-to-SQL translator, which generates one single SQL statement rather than multiple SQLs from a SPARQL query, and thus fully utilizes well-developed SQL execution engines. From the performance results we observe that the translator of the SMDM system greatly outperforms that of D2R server [1] in terms of execution time of the generated SQLs.

In this demonstration, with the IBM MDM model, we will highlight three main features in the SMDM system.

1. **Cross-domain data linking.** Data linking can be realized by 1) publishing various datasets as RDF data and 2) setting the links between the published data. Thus, by exposing master data as virtual RDF data, SMDM can link it with open data within an enterprise, and enables a unified customer view and product reorganization under different business domains.

2. **Dynamic concept extension.** The rich expressivity and formal semantics of the SMDM ontology model provide an explicit specification of conceptualization on business entities. According to personalized applications, customers can dynamically extend core concepts in master data by user-defined rules or OWL expressions [7] and apply them to business transactions.

3. **Semantic query and implicit relationship discovery.** With embedding reasoning into query evaluation, SMDM can discover hidden relationships between individuals in MDM. Semantic query integrates core MDM ontology, customer extensions, imported domain ontology, and business rules to derive new information from existing data.

In actual scenarios, it is usually easy for clients of SMDM without knowledge of implementation details to issue semantic queries and make declarative business definitions and policies by user-defined rules.

2. A TOUR OF THE SMDM SYSTEM

Figure 2. illustrates the architecture of the SMDM system. The extended SPARQL query is passed to the system from frontend users, and the SPARQL pattern tree is the key structure maintained by the SPARQL-to-SQL translator. We describe the details of four major components and our technical contribution.

Mapping component. Given the core MDM ontology, the system needs to build a RDB-to-ontology mapping over the master data hub so that a SPARQL query can be seamlessly translated into an executable SQL. Here we leverage the well-known D2RQ mapping language, a declarative mapping language for describing the relationship between an ontology model and a relational data model. With the D2RQ mapping tool and captured specific characteristics of the MDM relational model, we adopt a semi-automatic method to generate the mapping file.

Datalog rule engine. The datalog rule engine embraces its reasoning of two types: ontology reasoning and user-defined rule reasoning. Ontology reasoning is performed for finding implicit RDF triples within the ontology model. It is fulfilled based on Description Logic Programmes (DLP) [13], referring to a set of Datalog rules. For user-defined rule reasoning, users can define a set of business rules on top of ontology to enrich the semantics.

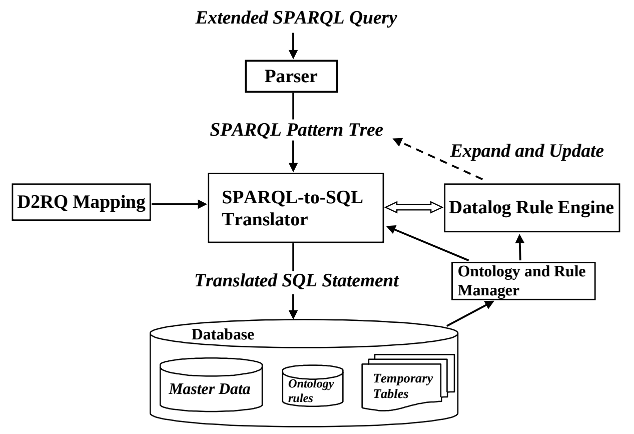


Figure 2: Architecture of the SMDM system

Considering the large size of master data and its update problem, it is not practical to materialize all the inferred results. We apply a runtime reasoning mechanism and follow the semi-naive evaluation approach with Magic Sets optimization [13] in the datalog evaluator.

One characteristic in the SMDM system is that instantaneous rule reasoning is injected into query evaluation. The high-level idea behind it is to expand the SPARQL pattern tree by the rule engine and update it if necessary. The rule engine first receives the parsed pattern tree, checks whether there are rule heads in pattern nodes and expands them by rule conditions accordingly. If there is a recursive loop when expanding a rule head, the datalog evaluator will evaluate the corresponding pattern node and generate a temporary table to store the inferred results. The rule engine then attaches the temporary table to the corresponding node and updates the pattern tree. The modified pattern tree will finally be translated into a SQL statement. The temporary tables, together with the underlying data tables, are involved in the final SQL execution.

We propose a TreeExpansion algorithm in this process. It constructs a dependency graph based on all the rules. In the pattern tree, if a node corresponds to recursive rules, it computes the strong connected subgraph it belongs to in the dependency graph. Then the datalog evaluator is triggered to compute fixpoints for all related rules in the strong connected subgraph. Otherwise (rules are not recursive), it simply expands the node as a subtree and recursively calls the TreeExpansion procedure on it.

SPARQL-to-SQL translator. We use our previous work [14] to express a SPARQL graph pattern as a node pattern tree, which contains five types of nodes, AND node, OR node, TRIPLE node, FILTER node, and N-ARY node.

Compared with the existing methods that translate a SPARQL query into a SQL, such as D2RQ and Virtuoso, the SMDM translator has the following highlights.

First, our method generates a single SQL statement rather than multi-stage SQLs so that well-developed SQL optimizers can be fully utilized. Besides standalone execution, the generated single SQL can also be directly embedded into other MDM SQL executor as a sub-query, which provides an easy way to seamlessly integrate normal MDM queries with semantic queries. The single-SQL generation relies on our Semi-SQL structure, which fills the gap between columns in

SQL and variables in SPARQL. Each pattern node is first translated into a Semi-SQL structure. Then all the Semi-SQL structures are flattened to reduce unnecessary joins and further translated into one SQL statement as a whole.

Second, the translator supports both the horizontal schema and the vertical schema[1], and makes optimizations on them respectively. A hybrid physical schema, including horizontal one and vertical one, appears common in many applications. To handle this, two additional components are added into the translator. One is the SPARQL Dispatcher, which decomposes the incoming query pattern into the horizontal part and vertical part according to extracted mapping information. The other is the SQL Coordinator, which combines generated intermediate SQLs from the horizontal part and vertical part into a final statement. We present a two-binding implementation in vertical translation, which utilizes the pattern-level ID binding rather than the in-pattern URI binding to greatly reduce join cost.

To examine the efficiency of the translator in the SMDM system, we build a set of benchmark queries in the testbed, which contain two kinds of queries, simple patterns and complex patterns. The former includes different combinations of variable/constant options on subject, predicate and object position in a triple pattern. The latter indicates the complex AND/OR patterns, which contain a number of different triple patterns. We conduct comparison on the retrieval time between the SPARQL-to-SQL translator of the SMDM system and that of D2R server. From the results, we observe that for simple patterns both of them have the similar retrieval time, while for complex patterns the former outperforms the latter with the retrieval time three to five times faster.

Ontology and rule manager. We apply the persistent mechanism to ontology and rule management. Two types of ontologies, including the core ontology and imported domain ontology, are stored in the SOR system we implemented [15], and a separate set of tables are used for storing rules. In execution, the translator and rule engine obtain the ontologies and rules they need from the manager.

3. DEMONSTRATION

3.1 Demo Scenario and Use Cases

To build the MDM database, we use the relational schema of Websphere Customer Center (WCC) [4], one of IBM's MDM products, and run the use cases on the data set of an insurance company. A pre-built WCC ontology, which is called the core MDM ontology, is loaded into the system beforehand and corresponds to key entities and relationships in the WCC model. The main entities in the data set contain 1.9M *Contract*, 1.0M *Claim*, 2.0M *Contact* including *Person* and *Organization*, and 3.2M *Location_Group*. It also contains 1.9M *Contact*-to-*Contract* relationships.

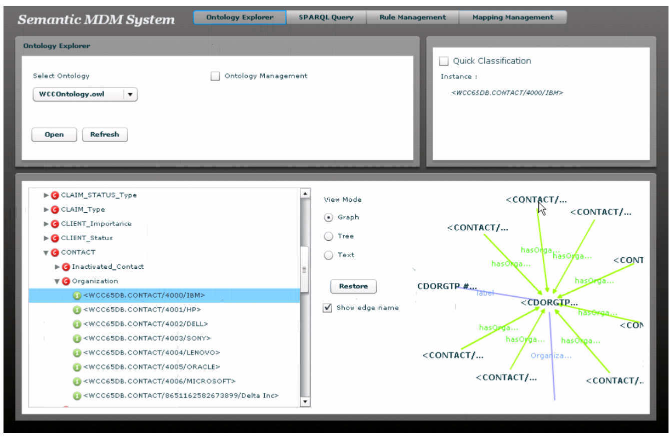
Motivated by actual market requirements, the insurance company tries to realize the following business points based on its existing capability.

1. The company wants to classify or reorganize their insurance products and services according to a formal standard in an external financial domain, which is different from its own product line domain. (*Cross-domain data linking*)

2. To improve the quality of personalized services, the company wants to introduce the concept of VIP customer, construct its explicit or implicit relationship with entities in the system, and finally identify a group of existing customers as VIP customers based on their personal and historical information. (*Dynamic concept extension*)

3. For preserving these VIP customers, the company will regularly provide them with special offers on its insurance products. The eligibility of the special offer is determined by sale promotion policies, which might involve some hidden relationships and should be flexible and subject to change. (*Semantic query and implicit relationship discovery*)

[1]Conceptually, the vertical schema involves a table with three columns (S, P, O), storing the values of subject, property and object in one row, while the horizontal schema is a general physical schema where a property corresponds to a column name.



Select Ontology: WCCOntology.owl
[ ] Ontology Management
[ ] Quick Classification
Instance : <WCC65DB.CONTACT/4000/IBM>

CLAIM_STATUS_Type
CLAIM_Type
CLIENT_Importance
CLIENT_Status
CONTACT
Inactivated_Contact
Organization
<WCC65DB.CONTACT/4000/IBM>
<WCC65DB.CONTACT/4001/HP>
<WCC65DB.CONTACT/4002/DELL>
<WCC65DB.CONTACT/4003/SONY>
<WCC65DB.CONTACT/4004/LENOVO>
<WCC65DB.CONTACT/4005/ORACLE>
<WCC65DB.CONTACT/4006/MICROSOFT>
<WCC65DB.CONTACT/8651162582673899/Delta Inc>

View Mode: [✓ Graph] Tree Text
[✓] Show edge name

Figure 3: A snapshot of the SMDM GUI

From the perspective of master data, the SMDM system captures the semantics-level association and provides a natural solution to satisfy these requirements, which correspond to the main features mentioned in Section 1.

Let's see what will happen when the insurance company follows the SMDM system. Firstly it imports an external financial domain (a hierarchy of classes), which is called the external domain ontology, to the system. With it the company can establish the RDF links between the product instances (in the form of RDF) and the imported domain classes, to reorganize the products in another customized line. We provide two ways to realize this from the user perspective. One is to select the instances by navigating to them and manually insert them into the target class. The other is to apply user-defined rules to virtually map a batch of products to a class in the imported domain hierarchy.

Secondly, by the company strategy, the VIP customer is defined as the one who either has the *clientImportance* label set to *highest* or is recommended by another VIP customer. We can see that this is a recursive definition and the datalog rule engine will be invoked to retrieve results. According to the existing core MDM ontology, business users without knowledge of implementation details can easily define such a concept VIP_Customer using rules, as is shown below.

```
VIP_Customer(?x):- wcc:hasClientImportance(?x, 'highest').
VIP_Customer(?x):- VIP_Customer(?y), wcc:providedBy(?x, ?y).
```

Note that a concept can also be defined using OWL expressions by advanced users, which will further be translated to datalog rules for evaluation.

Thirdly, the company defines the sale promotion policy as follows: a VIP customer who purchased an insurance product of class C and made less than 2 claims in previous years can get 20% discount for purchasing any C-typed products for the next year. Note that the policy implies a number of transitive *subClassOf* relationships, which take C as the root class and flow in the whole class hierarchy.

Users can construct rules to refine business concepts in a bottom-up way, and build semantic queries based on SPARQL extension. Take the imported class *Automobile* as an example. Given a customer <personX> and an *Automobile* insurance product purchased, the goal is to check whether the customer is eligible for the special offer. The corresponding rule-based extended SPARQL is descried as follows.

```
SELECT ?customer ?numOfClaims
WHERE {Contract_Owner(?customer, ?contract).
          Contract_Product_Relationship(?contract, ?product).
          ?product rdf:type import:Automobile. }
OPTIONAL {Contract_NumOfClaims(?contract, ?numOfClaims).}
FILTER {(?customer = <personX>)}
```

Contract_Owner, *Contract_Product_Relationship* and *Contract_NumOfClaims* are three extended concepts by rules, which define the contract owner relationship between a contact and a contract, the relationship between a contract and a product, and how to find a contract with the number of claims made from it. We omit their details due to space limitation.

Whether a customer is eligible for the special offer is examined in SMDM transactions. The special offer will go into effect automatically when the SMDM system detects that a VIP customer is eligible for it. In this example, if the result set is not empty and the returned *numOfClaims* is less than 2, <personX> can have 20% discount on the *Automobile* insurance product.

It should be mentioned that reasoning will be triggered several times in this process, because 1) the *subClassOf* relationship, implied in the policy, appears common in a class hierarchy. If class *Personal_Automobile* is a subclass of *Automobile*, a direct instance of *Personal_Automobile* also belongs to the type of *Automobile* and will be involved in any *Automobile*-related inference and evaluation through reasoning; 2) many extended concepts and relationships have to be expressed by recursive definitions. In the above example, *Contract_NumOfClaims* depends on the *Sub_Contract* relationship, which is a recursive definition and involved in runtime datalog evaluation.

3.2 User Interface Overview

We implement the SMDM graphical user interface using Adobe Flex and Java Servlet, and run the SMDM system on Apache Tomcat as a web service. As is illustrated in Figure 3, it is divided into four parts: Ontology Explorer, Semantic Query, Rule Management, and Mapping Management. By clicking an instance under a certain ontology class with tree-structured multi-level nagivation, users can continuously view all the linked data that is related to the instance under the star-style graph view mode. Users can also perform a series of flexible operations in the interface, such as defining rules to map business instances to external domain concepts, loading semantic queries from pre-defined templates and so on. Combined with the user interface, the demonstration will tell a whole story based on the above scenario and use cases in a straightforward way.

4. POTENTIAL APPLICATIONS

Recently we successfully applied the SMDM technology to the healthcare domain. We have built a platform that allows physicians to access clinical data through virtual RDF store and supports semantic queries and analysis based on the healthcare ontologies. The schema of clinical data is derived from the standard HL7 RIM [3] model and we build a general repository using IBM Clinical Genomics (CG) to store clinical data in a relational database. HL7 RIM ontology, considered as counterpart of the core MDM ontology in SMDM, covers all the aspects of clinical data, including clinical entities and rich relationships. By building the mapping between the relational schema in CG and the RIM ontology, we can expose the relational clinical data as virtual RDF data. We also introduce SNOMED CT ontology [10], which defines the formal semantics of clinical concepts and plays the similar role as what the imported domain ontology does in SMDM. By linking the clinical data (modeled by RIM) with the clinical concepts (modeled by SNOMED CT), we enrich the formal semantics of clinical data, and enable semantic queries and user-defined rules on it. As a result, implicit clinical information can be inferred and delivered to end users.

In future work, we would like to improve the consumption of the SMDM technologies, and integrate the SMDM system into the existing MDM solutions, to enrich more enterprise-wide applications.

5. REFERENCES

[1] D2RQ. http://www4.wiwiss.fu-berlin.de/bizer/d2rq/.

[2] DBpedia. http://dbpedia.org/About.

[3] HL7 RIM. http://www.hl7.org/Library/data-model/RIM/modelpage_mem.htm.

[4] IBM Websphere Customer Center. http://www-01.ibm.com/software/data/masterdata/customer/.

[5] Mapping relational data to rdf in virtuoso. http://virtuoso.openlinksw.com/wiki/main/main/vossqlrdf.

[6] Oracle Master Data Management Suite. http://www.oracle.com/master-data-management.

[7] OWL. http://www.w3.org/2004/OWL/.

[8] SAP NetWeaver Master Data Management. http://www.sap.com/platform/netweaver/components/mdm.

[9] Semantic Web. http://www.w3.org/2001/sw/.

[10] SNOMED CT. http://www.nlm.nih.gov/research/umls/Snomed/snomed_main.html.

[11] SPARQL Query Language for RDF. http://www.w3.org/TR/rdf-sparql-query/.

[12] K. Bollacker, C. Evans, P. Paritosh, and T. Sturge. Freebase: a collaboratively created graph database for structuring human knowledge. In *SIGMOD*, 2008.

[13] S. Ceri, G. Gottlob, and L. Tanca. Logic programming and databases. Springer-Verlag, 1990.

[14] J. Lu, F. Cao, L. Ma, Y. Yu, and Y. Pan. An effective sparql support over relational databases. In *VLDB Joint ODBIS-SWDB workshop*, 2007.

[15] J. Lu, L. Ma, L. Zhang, J. Brunner, C. Wang, Y. Pan, and Y. Yu. Sor: A practical system for ontology storage, reasoning and search. In *VLDB*, 2007.

[16] H. Morris and D. Vesset. Managing master data for business performance management: the issues and hyperion's solution. IDC white paper, 2005.