Querying SAP ERP with SPARQL

Marc Kirchhoff
SAP Research
Chemnitzer Straße 48
01187 Dresden, Germany
marc.kirchhoff@sap.com

Kurt Geihs
University of Kassel
Wilhelmshöher Allee 73
34121 Kassel, Germany
geihs@uni-kassel.de

ABSTRACT

Most of the mid-size and large companies employ enterprise resource planning (ERP) systems to manage their business data and business processes. Business entities like purchase orders, equipments or customers are represented as business objects in ERP systems. Multiple business objects can span a business object graph by referencing each other. The SAP ERP system provides several interfaces to access business objects. Querying business objects with regard to their positions within a business object graph by using these existing interfaces is not an easy task. In this paper, we propose to use the RDF query language SPARQL to express queries against business object graphs. We argue that SPARQL is well suited to query business objects in ERP systems and introduce an architecture to provide a SPARQL endpoint on top of an SAP ERP system. This novel approach simplifies the retrieval of data from ERP systems and makes it available to the integration with the Semantic Web.

Categories and Subject Descriptors

H.4 [Information Systems Applications]: Miscellaneous; J.1 [Administrative Data Processing]: Business

General Terms

Management

Keywords

SPARQL, SAP, ERP, Ontologies, Business Objects

1. Introduction

Enterprise resource planning (ERP) systems are widely employed by many companies to manage their business data and business processes. We provide a design for a SPARQL [7] endpoint on top of an ERP system to make the data available for the integration with the semantic web and to simplify the retrieval of the data. As SAP is the market leader in the ERP segment [2] we focus on the SAP ERP system.

Within ERP systems, business entities like purchase orders, contracts, materials etc. can be represented as business objects. Business objects hold a set of attributes (*name*, *description* etc.) and associations to other business objects. By relating each other several business objects can span a graph (business object graph). Figure 1 shows a simplified business object graph created within an ERP system as part of two sales order processes. If a customer requests a company to deliver goods, a sales order business object is created (e.g. *Sales Order 1A*). The information that is needed to execute the production process to build the goods is stored as a production order business object (e.g. *Production Order 1A*). Both business objects are related to each other. Several different types of materials are needed to start the production process. The corresponding business objects (*Material 1A*, *1B* and *1C*) are referenced by the production order business object. As there is a lack of material *1A* and material *1C* these are ordered from external suppliers. The orders are represented by purchase order business objects which are associated with the corresponding materials.

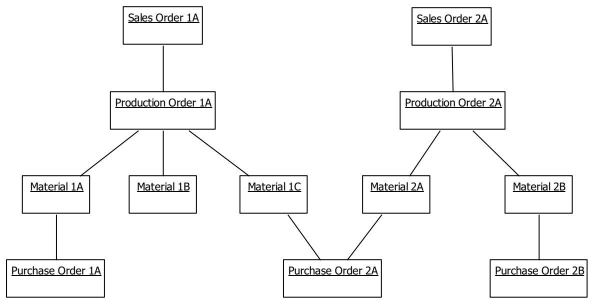

Figure 1: A business object graph.

If an external supplier is unable to fulfill a purchase order on time (e.g. purchase order *2A*) the start of the production processes that depend on the corresponding materials are at risk. The challenge is to determine the affected production orders (*Production Order 1A* and *2A*) and sales orders (*Sales order 1A* and *2A*).

The SAP ERP system already provides several interfaces to query business objects, e.g.:

- Business Application Programming Interface (BAPI): Proprietary, remote-enabled function modules.
- Enterprise Services (ES): A SOAP/WSDL-based web service interface.
- NetWeaver Gateway: Provides REST based access to business object data.

These interfaces provide a stable and standardized way to access the SAP ERP from non-SAP applications (see Section 2). The functions and services of the BAPI and the ES are grouped according to the business object types. For most of the business object types the interfaces provide basic functionality to query the business objects according to their properties. This makes it easy to query the ERP for a specific business object. However, the querying of information about business objects with regard to their relation to other business objects as described above is not an easy task. One of the reasons is that the functions provided by the interfaces usually do not allow queries spanning multiple business objects. Figure 2 shows a simple business object graph. An equipment with a specified serial number (*234867*) produced by the manufacturer *XYC* has several documents attached to it.

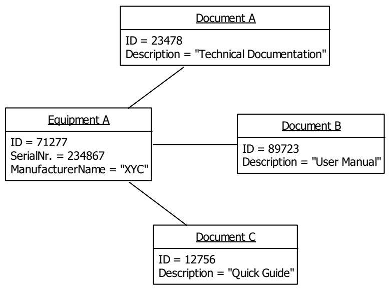

Figure 2: Documents attached to an equipment.

Neither the standard BAPI nor the Enterprise Services provide a function or a service to query the descriptions of the documents for the given serial number and manufacturer name of the equipment. Of course, it is possible to develop a BAPI function or an Enterprise Service that provides this functionality but this requires additional effort. We are only considering the standard interfaces provided as is. In this case the internal ID of the equipment (specified by the serial number and the name of the manufacturer) needs to be retrieved before the attached documents can be queried. Two different web service calls are necessary to retrieve this ID using the Enterprise Services. With the equipment ID another web service can be called to get the IDs of the attached documents. The descriptions of the documents can be queried by a service provided by the document business object using the IDs as input. This simple query involves only two business object types (Equipment and Document) but it requires 4 different Enterprise Services invocations to be executed. The usage of the BAPI would result in a similar complex sequence of calls. Queries that involve more business object types, for example the query described above (Figure 1), result in even more complex service orchestrations.

In order to simplify the retrieval of business objects with regard to their position within the business object graph we propose to use semantic technologies. The RDF query language SPARQL allows the expression of such queries in much more compact and intuitive way. Listing 1 shows the SPARQL query for the business object graph in Figure 2. Instead of almost 100 lines of code that are necessary to call the four different web service and filter the results, the whole query can be expressed as a single, very short SPARQL query. Another advantage is that the developer does not have to “guess” the relations between the different business objects because they are made explicit by the ontology. As SPARQL is a graph-based query language it is also better suited for querying business object graphs than SQL, which is based on the relational model. Specific queries can be expressed in a much more compact and intuitive way (e.g. if the length of the path between two nodes is unknown).

```
SELECT ?desc
WHERE {
 ?equ rdf:type boo:Equipment .
 ?equ boo:hasManufacturerSerialID 234867 .
 ?equ boo:hasManufacturerName "XYC" .
 ?equ boo:hasDocument ?doc .
 ?doc boo:hasDescription ?desc }
```

Listing 1: SPARQL query to request the document descriptions.

In this paper, we present an architecture for a semantic layer that provides a SPARQL endpoint for an SAP ERP systems. The remainder of this paper is structured as follows. In Section 2 we discuss why it is not a good idea to place the SPARQL endpoint on top of the ERP database. In Section 3 we present the architecture of our system. Related work is discussed in Section 4. Section 5 summarizes the paper and provides an outlook on future work.

2. Access method

In general, two different approaches exist to provide a SPARQL endpoint for an arbitrary data source:

- ETL (Extract Transform Load): The data is pulled from the data source, converted into RDF and stored in a triple store. The SPARQL processor operates on the RDF data stored in the triple store. Many different tools exists to convert various data formats to RDF
- Dynamic mapping: The SPARQL processor operates on the original data source. Only the data that is necessary to answer a given SPARQL query is dynamically retrieved.

Taking into account the large amount of fast changing data that is typically stored in an ERP system the replication of the data is not a viable option. Instead a more dynamic approach must be used. Business objects in the SAP ERP system are stored in relational databases. Many approaches and tools (e.g. Virtuoso or D2R server) are available to provide a SPARQL endpoint on top of a relational database. In order to standardize the different approaches a W3C Working Group (RDB2RDF) was founded. The working group currently focuses on the definition of a direct mapping from relational data to RDF [4] and on the specification of an RDB to RDF mapping language (R2RML) [6]. The direct mapping defines a simple transformation based on the schema of the relational database. Tables and table columns are mapped to RDF types and predicates. The triples are generated out of the table rows. Each table cell is converted into a triple. The subject IRI is automatically created based on the primary key value. The predicate is an IRI generated for the respective column in the table. The object is either the value of the table cell (mapped to the corresponding XML Schema data type) or a reference to another resource if the cell contains a foreign key. This approach can be used if the RDF vocabulary should directly reflect the schema of the relational database. However, in some cases a more customized mapping is needed. The RDB to RDF mapping language (R2RML) allows the definition of a mapping with regard to an arbitrary target vocabulary.

Due to the complexity and frequent changes of the internal ERP database schema neither of these two approaches can be used. As explained in Section 3, the ontology that is the foundation of the ERP SPARQL endpoint must model the business object types, their properties and their relations among each other. The direct mapping as proposed by the RDB2RDF Working Group cannot be used because the business object types usually do not have a direct mapping in the database schema. Instead business objects are often distributed across several tables. These tables and their columns which represent the properties of the business objects often have none-descriptive names. Hence, the direct mapping approach would result in a very complex ontology with several concepts for most of the business object types. Although it is possible to define a customized mapping by using the R2RML this solution has several drawbacks as well:

- The ERP database schema consists of several thousands of tables which are often difficult to understand. This makes it very complicated to define the mapping.
- The fact that the business object types usually do not have a direct mapping in the database schema makes it even more difficult to define the mapping.
- The database schema changes over time, i.e. it is not unusual that a new version of the ERP system has a different database schema than the previous one. Thus, a customized mapping based on the database schema would be valid only for a specific version. An adaption would be necessary for each new version.

Due to these reasons it is practically not feasible to manually define a mapping based on the database schema. A better solution is to place the SPARQL endpoint on top of existing ERP interfaces that allow the direct retrieval of business objects. This has several advantages:

- The interfaces are far less complex than the database schema and they are well documented which simplifies the definition of the mapping.
- The functions and services of the interfaces are assigned to business objects. Furthermore, they allow the direct retrieval of the data of the business objects. This simplifies the mapping to the corresponding business object concepts of the ontology.
- A mapping defined with regard to a specific ERP version will be valid for many future versions as downward compatibility is guaranteed for a long period of time.
- Functions and services released with a new version can be made available to the SPARQL endpoint easily by just adding a new mapping definition without changing the old one.

The SAP ERP provides several interfaces to access business objects. We considered the following: Business Application Programming Interface (BAPI), Enterprise Services (ES) and SAP NetWeaver Gateway. The BAPI is a set of remote function call (RFC) modules that are grouped according the business objects. BAPIs can be called from many different programming environments, e.g. ABAP, Java (SAP Java Connector), .NET (SAP Connector for Microsoft .NET). The Enterprise Service interface is a SOAP/WSDL-based Web Service interface. Currently, the interface provides services for over 330 business objects of the SAP ERP system. New Enterprise Services are added if a business need is identified by the Enterprise Services Community. Downward compatibility is guaranteed for the Enterprise Services as well as for the BAPI. SAP NetWeaver Gateway is a new development framework that can be used to expose business object data as REST based services. Currently, it does not provide as many predefined services as the BAPI or the Enterprise Services and it is not available on many ERP systems. Therefore, we build our system on top of the Enterprise Services.

3. Architecture

In this section we describe the main components of the system. Figure 3 shows the overall architecture of the semantic ERP layer (S-ERP Layer). The layer provides a SPARQL endpoint on top of the ERP system. It can be used by client applications to query the ERP system.

A SPARQL query sent by the client is received by the Orchestration Engine (OE) where it is transformed into an internal data model. Based on the syntactic and semantic descriptions of the Enterprise Services stored in the Service Registry the Orchestration Engine determines the Enterprise Services that need to be called to answer the query. The OE sends the orchestration information to the Execution Engine which calls the corresponding Enterprise Services. The result is either forwarded directly to the Orchestration Engine which returns it to the client or, if reasoning is needed, it is forwarded to an RDF Adapter.

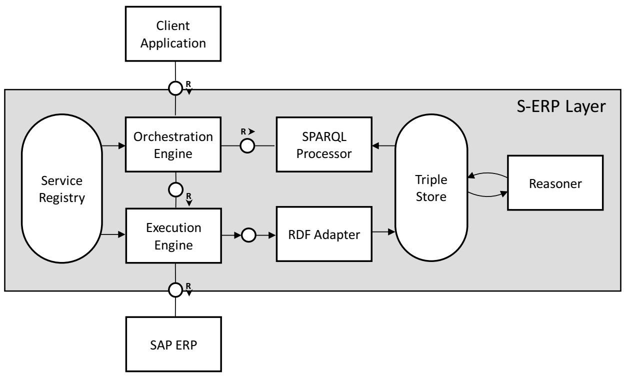

Figure 3: S-ERP layer architecture.

The RDF adapter transforms the business object data into the RDF data format. Rule-based reasoning is provided by a reasoner that operates on the RDF data stored in the triple store. The actual SPARQL query sent by the client is executed by the SPARQL processor against the data in the triple store.

The ontology is the central component of the whole system as it determines the data that can be queried from the ERP system. It must reflect the business objects, their attributes and associations to other business objects. However, as the S-ERP layer uses the Enterprise Services to retrieve the data from the ERP system the ontology has to be aligned with the data model used by the Enterprise Services. This data model is based on the Global Data Types (GDTs) [9]. The GDTs are SAP-wide standardized data types derived from the Core Data Types as specified in the UN/CEFACT Core Component Technical Specification (CCTS) [1]. They are described in XML Schema and directly used by the Enterprise Services. Figure 4 shows the relation between the

GDTs, the business objects and the Enterprise Services (Service Operation). The ontology must reflect this structure. We currently investigate methods to automatically create the ontology out of this data model and to automatically semantically annotate the Enterprise Service descriptions.

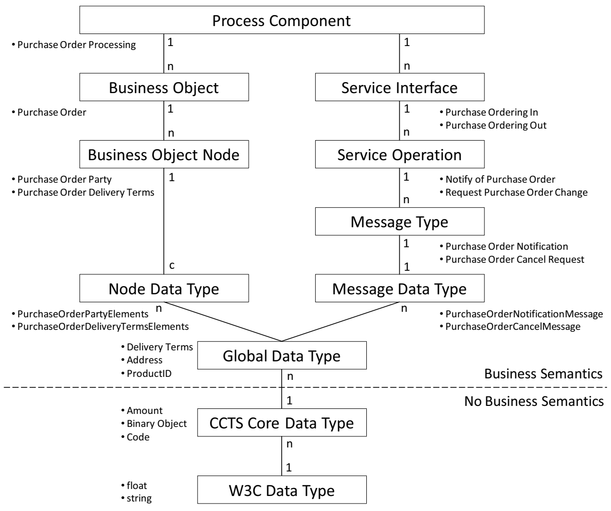

Figure 4: An overview of the ES meta model [9].

4. Related Work

Several attempts have been made internally at SAP to expose ERP business data as RDF data. Due to the complexity and structure of real SAP ERP databases most of these approaches rely on the replication of data. The ERP data that should be provided as RDF is first copied into a separate database. Existing mapping languages [4][6] and tools (e.g. D2RServer) are used to provide an RDF view for these databases. This approach has several disadvantages. The replication of the data requires an additional database that has to be installed on the customer side. Due to the large amount of data stored in real ERP systems the replication of the entire database is usually not feasible. As explained in Section 2, due to the complexity and frequent changes of the ERP database it is not viable to provide a SPARQL endpoint on top of the ERP database either.

There have been several approaches to integrate Web Services into the semantic web [3] [8] [5]. Lanthaler and Gütle [8] propose an approach to integrate web services into the Linked Data cloud by mapping SPARQL queries to HTTP requests. In contrast to our approach, they focus on RESTful data services using JSON as serialization format. Alcaron and Wilde focus on RESTful services as well [3]. They present a description language for RESTful Web Services (ReLL) and they show how RDF data can be extracted from RESTful services. The Semantic Bridge for Web Services (SBWS) presented by Battle and Benson [5] is a tool that provides a SPARQL endpoint for existing REST-based and SOAP-based web services. The tool is proprietary and many functions (e.g. the orchestration algorithm) remain unclear. Furthermore, it does not support SPARQL Update.

5. Conclusion and Future Work

In this paper, we presented and discussed an architectural approach for SPARQL as a query language for ERP systems. Compared to existing ERP interfaces SPARQL can greatly simplify the way data is queried from ERP systems. We presented an architecture for a SPARQL endpoint on top of an SAP ERP system. This approach will not only reduce the complexity of querying business objects from SAP ERP systems but also make ERP data available for the integration with the semantic web.

We have developed a proof-of-concept prototype that shows the feasibility of our approach. However, this is ongoing work. We are currently investigating methods to automatically create the ontology out of the ES-/ERP-data model and to simplify the semantic annotation of the Enterprise Services. In future work, we plan to cover more SPARQL features.

6. Acknowledgment

The work presented in this paper was supported by the RES-COM project funded by the Federal Ministry of Education and Research.

7. References

[1] Core components technical specification version 3.0. Tech. rep., United Nations Centre for Trade Facilitation and Electronic Business, September 2009. http://www.unece.org/fileadmin/DAM/cefact/codesfortrade/CCTS/CCTS-Version3.pdf.

[2] 2011 erp report, 2011. Panorama Consulting Group. http://panorama-consulting.com/resource-center/2011-erp-report/.

[3] ALARCON, R., AND WILDE, E. Linking Data from RESTful Services. In *Third Workshop on Linked Data on the Web (LODW2010)* (2010).

[4] ARENAS, M., BERTAILS, A., PRUD'HOMMEAUX, E., AND SEQUEDA, J. A direct mapping of relational data to rdf. W3C candidate recommendation, W3C, Feb. 2012. http://www.w3.org/TR/2012/CR-rdb-direct-mapping-20120223/.

[5] BATTLE, R., AND BENSON, E. Bridging the semantic web and web 2.0 with representational state transfer (rest). *Web Semantics: Science, Services and Agents on the World Wide Web 6* (February 2008), 61–69.

[6] DAS, S., SUNDARA, S., AND CYGANIAK, R. R2rml: Rdb to rdf mapping language. W3C candidate recommendation, W3C, Feb. 2012. http://www.w3.org/TR/2012/CR-r2rml-20120223/.

[7] HARRIS, S., SEABORNE, A., AND PRUD'HOMMEAUX, E. Sparql 1.1 query language. W3C working draft, W3C, Jan. 2012. http://www.w3.org/TR/2012/WD-sparql11-query-20120105/.

[8] LANTHALER, M., AND GUTL, C. Aligning web services with the semantic web to create a global read-write graph of data. In *Web Services (ECOWS), 2011 Ninth IEEE European Conference on* (sept. 2011), pp. 15 –22.

[9] SEUBERT, M., RICHTSTEIGER, D., REIDL, M., AND WOLF, S. Data type catalog - definitions of global data types and core data types. Tech. rep., SAP, Mar. 2012. http://www.sdn.sap.com/irj/scn/index?rid=/library/uuid/303fd192-1db2-2a10-59b9-9dfd93f4b10f&overridelayout=true.