International Journal of New Computer Architectures and their Applications (IJNCAA) 4(2): 56-69
The Society of Digital Information and Wireless Communications, 2014 (ISSN: 2220-9085)

Using Semantic Web Technologies in Heterogeneous Distributed Database System: A Case Study for Managing Energy Data on Mobile Devices

Zhan Liu, Anne Le Calvé, Fabian Cretton, Nicole Glassey
Institute of Business Information Systems
University of Applied Sciences and Arts Western Switzerland, Sierre, Switzerland
{zhan.liu, anne.lecalve, fabian.cretton, nicole.glassey}@hevs.ch

ABSTRACT

Managing energy data from multiple distributed and heterogeneous sources is an important issue worldwide. This study focused on using semantic web technologies in an energy data management system. Specifically, we provided a federated approach - a mediator server - that allows users to access to multiple heterogeneous data sources, including four typical types of databases in energy data resources: relational database, Triplestore, NoSQL database, and XML. A proposed architecture based on this approach is then presented and our solution can realize the data acquisition and integration without the need to rewrite or transform the local data into a unified data. We further examined our architecture by a case study in a Swiss energy company and tested the system based on a mobile platform.

KEYWORDS

Semantic web, heterogeneous distributed database system, SPARQL, RDF wrapper, database integration, energy management, mobile application

1 INTRODUCTION

Along with the contributing global warming, social concern over environmental problems has increasingly become a hot issue in recent years. Companies must systematically manage energy use and handle as much energy information as possible to get deep and quantitative knowledge of the process of energy consumption [1]. As an important part of information resource, energy information resource supports energy efficiency and influences the direction of future performance. Reducing energy consumption in the residential sector is important as well, because energy demand in this sector is notably increasing recently [2].

However, such information processing is not managed by an integrated database where all of the data relevant to an organization would be stored and managed in one single unified and integrated database. Rather, databases are non-integrated, distributed and heterogeneous [3]. This is especially evident in the context of an energy database management system due to several reasons. First and foremost, an energy database management system was built according to the characteristics of the energy usage in a specific region. Thus, the requirements are diverse and as a consequence, database systems in the field of energy are rather distributed and complicated. For example, electricity consumption function requires its information system to quickly convert and store a large body of non-relational data; thus, a NoSQL database like MongoDB [4] fits very well in this case. However, in other functions where the structure of the data is stable with low variability, and if such data have relationship with other data in the same data source, a relational database should be a good choice. Moreover, Triplestore is selected if the consumption data are necessary to integrate with other remote data resources, like geographic and weather information systems. In the case of a semi-structured data model, XML serves well and it is usually used to store and exchange information of configuration for different systems. Second, an integrated system was not the main goal at the time the database

systems were built [3]. Third, energy database systems that differ from each other may be caused by changes in technology. Last but not the least, in contemporary urban environments and at a household level, energy management requires that the design of systems be able to integrate remote and spatially distributed monitoring data while being open, low cost, easy to use and flexible [5]. All these characteristics indeed set barriers to getting accurate energy information in a global perspective. Only using the existing tools cannot solve these problems.

Nowadays, the increasing globalization has encouraged waves of mergers and acquisitions, presenting new difficulties for companies to handle huge amounts of complex and disparate information across regions [6]. Simply exchanging basic information today may involve accessing and interpreting a wide variety of formats, data language, data models, and protocols that go beyond just text. To achieve coordination of diverse computerized operations, it is necessary for a company to have database systems that can operate over a distributed network and can encompass a heterogeneous mix of hardware, operating systems, local database management systems and even data models for different databases. Consequently, integrating and querying data from heterogeneous sources has become a hot research topic among information researchers. In general, there are two possible approaches to the architecture of a heterogeneous distributed database: namely warehouse approach (e.g., [7]) and federated approach (e.g., [8]). The separation is sometimes called centralized and decentralized systems. The first method typically provides a uniform interface to materialize the integrated view. The latter approach, on the other hand, is a form of virtual integration – the data are brought together as needed [9]. In this study we focus on the federated approach, as under this architecture local databases can continue their local operations and transactions without changing the features of local databases; but at the same time participate in the federation. Therefore, this approach is more stable and reliable in our case.

The burgeoning semantic web technology has provided new methods for integration of heterogeneous distributed database management systems. According to Tim Berners-Lee et al. [10], the Semantic Web "provides a common framework that allows data to be shared and reused across application, enterprise, and community boundaries." While rapidly evolving, it is only recently that semantic web technologies are becoming available and stable, and practical solutions emerge and flourish in many fields. The idea involves the concept of Linked Data, which aims at enabling the same kind of possibilities for data, as well as creating a universal medium for exchanging information based on the meaning of content on the Web [11] in a way that is usable directly by machines. Resource Description Framework (RDF) is a general language to describe resources, especially on the web, and SPARQL is a query language for RDF that can join data from different databases, as well as documents, inference engines, or anything else that might express its knowledge as a directed labeled graph [12].

To this end, we proposed a uniform approach for SPARQL querying a heterogeneous distributed database system in energy data management. This federated method provides transparent query access to multiple heterogeneous data sources, including relational database, Triplestore, NoSQL database and XML, thus realizing the process data acquisition and integration without the necessity to rewrite or transform the local data. Most extant studies in heterogeneous distributed database systems only consider a single language (e.g., [3]) or only focus on relational data (e.g., [12]). Our approach is different from them in two ways: on one hand, we do not only look at one specific database model (e.g., relational database), but also provide solutions to integrate other database models. On the other hand, our mediator server does not require local databases to translate or transfer to a unified language; instead, all local sources remain at the original level thus it has cost advantages. Moreover, our solution fulfills the energy information calculation from the integrated data: e.g., daily, monthly, quarterly, etc.

This study aims to provide a better understanding for a heterogeneous distributed database system in

energy data management. More precisely, it contributes by:

(1) Proposing architecture for SPARQL querying a heterogeneous distributed database system. This system includes multiple heterogeneous data sources, such as relational database, Triplestore, NoSQL database and XML.

(2) Implementing the system according to the proposed approach by using semantic technologies. It also performs query optimization to speed search executions.

(3) Testing the proposed approach by a real case in energy sector in Switzerland.

(4) Designing and developing the system based on a mobile HTML5 platform. The use case gives a complete implementation of the mobile supported environment in which users are assisted about the energy consumption with the mobile application.

The remainder of this paper is structured as follows: we start with a discussion of related work in Section 2. Section 3 describes the architecture for a heterogeneous distributed database system, which consists of four layers, namely distributed data layer, RDF wrapper layer, mediator layer and user interface layer. Section 4 discusses the mediator layer implementation. The case study based on a Swiss energy company and a mobile application based on our architecture is presented in Section 5. Section 6 provides an evaluation and discussion of our solution. We conclude with a description of ongoing and future work in Section 7.

2 LITERATURE REVIEW

2.1 Energy Information System

Many information technologies have been introduced in managing energy information. As a result, a number of energy information systems have been proposed and used to monitor the sources of energy cost by identifying the energy consumption, then providing further analysis, and finally achieving energy effectiveness and efficiency (e.g., [13]; [14]; [2]), both on the industrial level (e.g., [15]; [1]) and resident level (e.g., [2]). For example, the building sector is reported as responsible for 40% of the total European energy consumption and 36% of the European Union's total CO2 emission [16]. But at the same time, it offers technological opportunities to improve energy efficiency and greenhouse gas emission. Accordingly, a number of researches have been conducted. For example, [17] established an integrated control system to optimize energy consumption and user comfort in buildings, [18] investigated and proposed a central control management system to optimize of electric energy consumption operating costs in cooling technique, and [13] developed a system to monitor building energy consumption for large public and governmental buildings. The network layer adopts the RS485 protocol, though it was argued by [1] that the RS485 has "too many limitations to be adopted in the multiple information or process control systems" (p. 489).

However, as energy information is rather diverse and consequently leads to different information systems, along with rapid development of information technologies, the information system in the energy sector has become increasingly complicated. In fact, the systems used are rather independent and heterogeneous, resulting in an "information island" and making it difficult to get accurate and transparent energy information quickly [1]. In the next session, we will briefly discuss the characteristics of heterogeneous distributed database system.

2.2 Heterogeneous Distributed Database System

Data discovery, date mining, and date integration have been important research topics in the field of heterogeneous distributed database management systems for years. Two possible approaches are briefly described as follows:

The warehouse and RDFizer [19] approaches usually consolidate data from multiple sources. The advantages of this method are the high efficiency and the capability of extracting deeper

information for decision making [1]. However, the warehouse database must set up the "data cleansing" and "data standardized" areas before its actual use. Overlapping and inconsistent information may exist among local sources; thus, it must be cleansed. Moreover, each local database may adopt different models from the warehouse's (e.g., schema, data type); thus, local sources need to be reshaped and transformed into a common one, that is, "data standardized". As a consequence, it typically would take months of planning and effort to create [9].

In contrast, the federated approach provides a single interface to many underlying data sources without the user explicitly specifying the target data source in the query. The advantages of data federation are the high adaptability to frequent changes of data sources, and the support of large numbers of data sources and data sources with high heterogeneity [1].

A large of variety of federated queries has been proposed recently for heterogeneous distributed databases (e.g., [20]; [21]). SPARQL, as a query language for RDF, has been well accepted to support querying of multiple RDF databases. It aims to find matching resources from a graph-like connected web for the database community [22]. For example, both [23] and [24] described the approaches for SPARQL queries over a catalogue of remote endpoints from multiple distributed relational databases. Moreover [25] introduced a SPARQL query mechanism for mapping relational databases to an ontologies approach. Contrary to other approaches, they took the complete schema of the database into account, creating a database specific ontology. To the best of our knowledge, no existing research addresses SPARQL federated query to support a heterogeneous distributed database system including the most current and popular databases, such as relational, Triplestore, NOSQL, and XML. It provides transparent query access over mapped RDF data sources. Our approach offers a standard SPARQL query interface to retrieve the desired distributed data in RDF format.

Most studies on SPARQL query optimization for a heterogeneous distributed database system include two aspects: minimizing communication cost and optimizing execution localization. According to [26], communication cost is reflected in the number of contacted data sources. It directly influences the performance of the query execution due to the communication overhead. The approach of query rewriting identifies the complex elements and proposes specific rewriting rules; therefore, it could be used to resolve the cost of communication among different databases ([27]; [28]; [29]). From [26] point of view, optimizing execution localization is represented by identifying optimal index structure and join ordering in order to execute queries in parallel and reduce query execution time. However, there is still little discussion of SPARQL query optimization across multiple heterogeneous databases. We aim to fill this research gap.

3 ARCHITECTURE OF HETEROGENEOUS DISTRIBUTED DATABASE SYSTEM

In this section, we first present a global view of our architecture; then, we will introduce in detail each component in the heterogeneous distributed database system for energy management.

The architecture, as shown in Figure 1, contains four principal layers. The first layer includes users' interfaces on mobile devices and in it, user could send one or multiple queries via mobile interfaces to the Mediator layer. The mediator is on the second layer of our architecture. It is a middleware system containing a global schema that describes the data throughout the network, and it is used to support and coordinate the distributed transaction management. The mediator is designed to integrate any kind of component database. Four important components are stored in this layer: Query Parser, Distributed Query Decomposer, Query Optimizer, and Transaction Coordinator. Once a user's query is received by the mediator, the query will be scanned and parsed into a graph structure of SPARQL. If no error is found, the generated transactions corresponding to the query are sent to the Distributed Query Decomposer, which can interpret the query

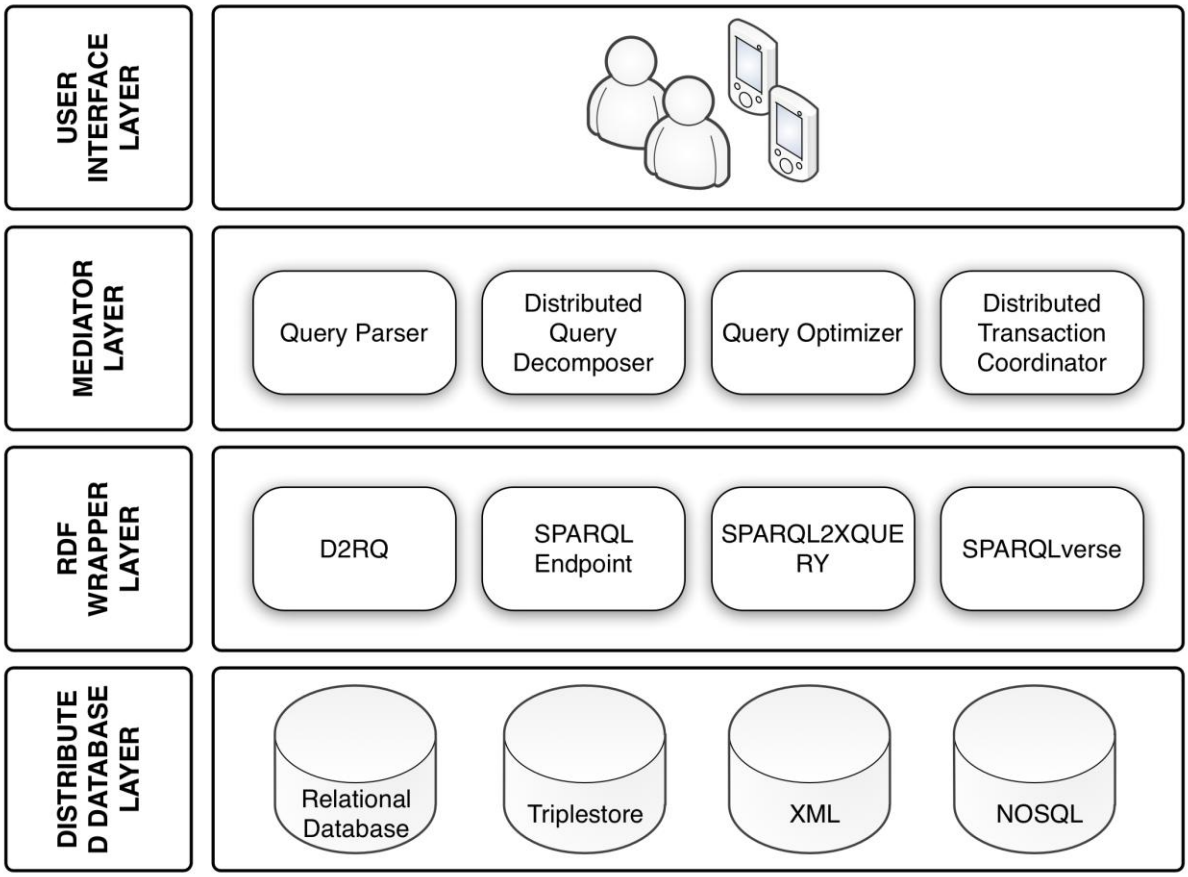

Figure 1. The architecture of heterogeneous distributed database system

received from the user's interface and generates a distributed query context containing several transactions and their associations (i.e., joins) [30]. Then, Query Optimizer takes all distributed transactions, and generates optimal sub-queries to build an optimal SPARQL query execution plan. The optimization of such sub-queries is a key factor concerning the performance of the overall system. For each distributed transaction, the Distributed Transaction Coordinator looks up the corresponding distributed database schema at which the accessed relation of the transaction resides from the definition of endpoint addresses. However, SPARQL queries require an explicit definition of endpoint URIs. Our system allows execution of queries without the necessity to specify target remote endpoints. After that, the Transaction Coordinator generates the navigation information in the form of SPARQL for all sub-transactions according to the associations among them, and sends then separately to the related heterogonous database server. The third layer of our architecture contains four different RDF wrappers such as D2RQ [31], SPARQL endpoint, SPARQL2XQUERY [32] and SPARQLverse [33]. These free and open RDF wrappers exactly corresponding four heterogeneous databases in the fourth layer - distributed database layer: relational database, Semantic Web Triplestore, XML, and NOSQL database. In order to encapsulate the details of component databases, these RDF wrappers are associated and placed on the top of distributed database layer. Therefore, when a SPARQL query arrives at the heterogonous distributed database servers, the query does not directly refer to distributed database. Instead, it contains graph patterns adhering to a virtual RDF data set. In addition, RDF wrappers also participate in query optimization. Then, the corresponding RDF wrapper generates the SPARQL query into a local query to the local schema DBMS. The execution result of the local

transaction is returned back to the same wrapper and then, the local result is converted to a uniform format (e.g., XML or JSON) and is collected by the Mediator layer. Finally, the client receives the global results in the form of HTML5 on their mobile phone's interfaces.

Now, we will present each component in our architecture in details as follows:

3.1 Query Parser

Query parser is used to scan and parse query statements to check syntactic errors, such as query references, names of relations, and attributes.

3.2 Distributed Query Decomposer

Distributed query decomposer generates a number of transactions to match the underlying remote data sources. These distributed transactions are submitted and executed in parallel with heterogeneous databases over remote connections. Moreover, the distributed query decomposer assembles transaction results and returns a final result to the end user.

3.3 Query Optimizer

SPARQL query optimizer in mediator layer provides an approach of the query execution plan to minimize the communication and processing costs to transmit query and result between mediator and heterogeneous distributed databases. In fact, the join order has a significant influence on the cost-effective query execution plan. Therefore, the join order optimization is usually the main focus of SPARQL query optimization. In our architecture, we proposed two steps for query optimization, namely data source optimization and join order optimization.

The data source optimization is represented by the precision of the data source selection and building sub-queries. The idea is to determine all return results from different data sources. Specifically, the data source selection would identify whether the return result to the SPARQL query is empty and which data source does not need to be accessed. Therefore, we send SPARQL ASK queries [34] including the triple pattern to all the federation databases and eliminate sources that fail to match the pattern. This refining of data sources is more efficient than accepting no results in regular SPARQL SELECT queries. The results from source selection are then used to build sub-queries. Each sub-query contains triple elements: triple patterns, value constraints and data source that can answer the sub-query. One sub-query could be matched to one or multiple data sources. The join order optimization is implemented in our solution to determine the numbers of intermediate results, since all query execution plans for heterogeneous distributed databases are based on the sub-queries generated by the data source selection. It is possible to use sub-queries joining larger result sets in a nested loop join after the smaller result set has been received completely. This mode of join order is called "Mediator Join" [26]. It executes the joins in the mediator after comparing the intermediate result sets from the data sources, and only the smaller results set will be returned to the mediator. This approach of join order is used in our query optimizer to deal with large result sets and it will drastically reduce the transfer costs.

3.4 Distributed Transaction Coordinator

A distributed transaction coordinator is used in our architecture to manager optimal distributed sub-transactions. It detects and handles persistent records of the transactions, and manages the communications with the databases.

3.5 SPARQL Endpoint

A SPARQL endpoint allows users to query a machine-friendly interface towards a knowledge base such as triple store via the SPARQL language. The results are returned in machine-processable formats, like XML and JSON. In our case, the four different RDF Wrappers in the

distributed database server could be considered as four SPARQL endpoints.

3.6 Mapping Relational Data to RDF

There are existing approaches for mapping relational data to RDF, such as Triply [35], R2O [36], and RDBToOnto [37]. In this study, we chose the approach of D2R to ease integration of our relational database and discover information without replicating the data into a dedicated RDF triple store. The D2R server uses D2RD mapping language to provide an automated process to generate the mapping file between specific relational database schemas and RDF schemas. This mapping file convers all tables from relational database to RDF classes, and it is used to identify resources, as well as access and generate property values into RDF format from database content.

The D2R server allows applications to query relational databases using SPARQL query language through the SPARQL protocol. Once the SPARQL requests arrive from the mediator, they are rewritten into SQL queries via the mapping and executed against a D2RQ-mapped relational database. Finally, the query results will be represented in XML and JSON formats and integrated into global results.

3.7 Mapping NOSQL Data to RDF

Several studies have been conducted of NOSQL databases and relational databases in the past couple of years. However, these studies focused mainly on the conversion between these two formats and which type of database is more effective and optimized for specific database management issues. Until now, little attention has been paid to the integration of NOSQL data and RDF. SPARQLverse is one of few tools that could help map NOSQL data to RDF. SPARQLverse is developed to meet W3C standards for the RDF/SPARQL 1.1; therefore, it could be used as an RDF database. In fact, SPARQLverse is an in-memory triplestore that provides the ability to connect to NOSQL databases such as MongoDB for data storage. Similar to D2R for relational databases, the SPARQLverse server provides an automatic mapping mechanism and it allows query graph style linked data and document-based data in MongoDB by using SPARQL. This mapping mechanism provides a “read-only” mode for the database content; thus, any request of adding, updating, and deleting from the user will not change any data in MongoDB. When a SPARQL request is sent to the SPARQLverse, the leader node dedicates a set of threads to process the request in parallel. Before sending the stream and starting processing, three steps are executed: (1) parse the initial query tree for the planner; (2) plan the steps for the query requirements; (3) generate all the segments for the query into a stream. Finally, the nodes return the results in JSON formats.

3.8 Mapping XML Data to RDF

XML has been widely successful for configuration information storage and information exchange on the Web. XML defines a set of rules to describe the content of structured and unstructured documents in a format that is both human-readable and machine-readable [38]. Several research works have clearly observed striking similarities between semi-structured data models and XML ([39]; [40]). These similarities are reflected in their irregular or often changing structure, as well as different attributes for different entities represented in a model that is often based on using tree or graph data structures.

A number of combinations of Semantic Web and XML technologies have been exploited. However, the objectives of these research works (e.g., [41]; [42]) only focus on data transformation from XML to RDF. SPARQL2XQuery represented a comprehensive framework that allows expressing semantic queries on top of XML data through the translation of SPARQL queries in XQuery syntax. SPARQL2XQuery proposed two types of scenarios to query CML data by using SPARQL. The first scenario is based on an automatically generated mapping ontology, and the second is

based on an existing OWL ontology. In our framework, the first scenario is matched and used to generate the mappings between the ontology and the XML schema automatically, as well as to integrate and query the XML data from the Sematic Web environment. The query results are transformed into the desired formats (such as XML or RDF) and returned to the mediator layer.

4 MEDIATOR LAYER IMPLEMENTATION

Once the heterogeneous systems (distributed database layer) and their SPARQL endpoints are set up (RDF wrapper layer), the mediator layer provides a uniform way to query those different endpoints with SPARQL.

To manipulate RDF data, semantic web toolkits exist for most programming languages (e.g. C++, Java, PHP, Python). Developers in the Java world have been very active in this field since the early days of RDF. They provide, amongst others, two famous frameworks called Jena [43] and Sesame [44], both of which support interaction with most of the currently available triplestores. According to [45], Jena offers faster load times and better scale, but provides worse query performance than Sesame. Because our study focused more on query performance, we decided to build our architecture on Sesame.

In our distributed system, we rely on the SERVICE extension. It has become a recommendation of SPARQL 1.1 federated query [46] since 2013. This extension allows us to direct a portion of a predefined query to a specific SPARQL endpoint. Therefore, we can write queries that request data from different databases through their RDF wrappers. The URL of the wrapper is the endpoint specified for the SERVICE:

```
SELECT * WHERE {
?building a musyopOnto:Building;
musyopOnto:hasID ?buildingID ;
SERVICE <http://d2r_server/sparql> {
?reading a db_vocab:readings;
db_vocab:reading_building_id ?buildingID
}}
```

The query shown above described how data can be merged from two sources: a building (`?building`) and its identifier (`?buildingID`) queried from the triple store, whereas the readings about this building (energy consumptions captor's readings) are queried from the relational database.

The example further illustrates the D2RQ mapping between the database tables and the RDF representation. Such mapping would tell D2R to translate the SPARQL query to SQL by matching the underlying table through the predefined `db_vocab:readings` mapping.

Moreover, to effectively handle the combination of result sets (as SQL joins) in this context of distributed but interdependent data sources, the solution makes good use of the VALUES keyword that allows filtering results by providing a table with values. It is thus possible to restrict the scope of subsequent sub-queries according to the results of previous ones, even though the results are directed to different endpoints.

5. CASE STUDY: ENERGY DATA MANAGEMENT IN SWITZERLAND

5.1 Overview of Energy Management in Switzerland

Swiss energy production is dominated by hydro power (56%) and nuclear energy (39%). This country consumed more than 61800 gigawatt hours of electricity and produced a net total of 63900 gigawatt hours. Because the energy sector's requirements for information system are diverse and information technology is developing rapidly, discussion about only one database is not enough, and the heterogeneous system is a must. Our solution aimed at offering as heterogeneous distributed system that can manage energy data in a rational and efficient manner but keep the cost at a relatively low level.

5.2 Case Study

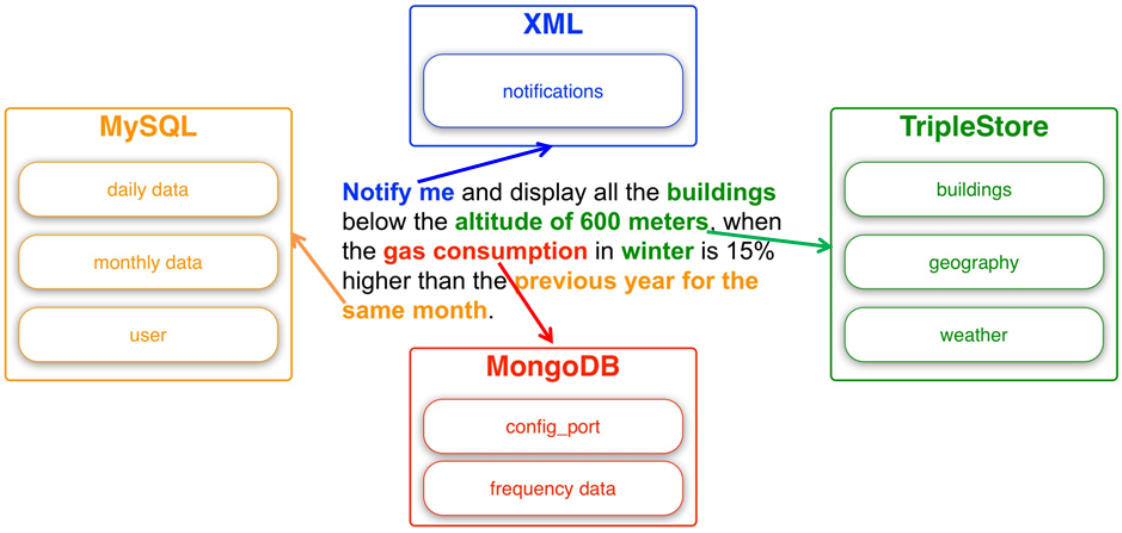

Figure 2. Case study with the related databases

To test the performance of our architecture, we designed and implemented a real case regarding energy data management. The case is based on issues that a Swiss energy company is currently facing. By using its real-time databases of electronic and gas consumption for a number of residents, our case study will illustrate the capability of our heterogeneous distributed database system to retrieve information from energy consumption in the optimization of data flow.

The existing databases include four different types of energy data that have been stored into four different databases in the distributed system according to their features. For instance, the real-time frequent energy data of consumption (e.g. minutely, secondly) is stored in MongoDB, because MongoDB fits very well to convert and store a large body of non-relational data. Users' notification settings are stored in XML files. The information related to geography, buildings, and weather is stored in Triplestore because the semantic data is easier to integrate with other remote resources on the Web. Moreover, the ontologies like geonames and weather already have been created and are widely used. Using semantic data allows us to access existing ontologies easily. All the aggregation energy data from MongoDB have been stored in a MySQL database to highlight the relationship between energy consumptions and time periods, for example, daily, monthly, and yearly consumption data.

Our case study described on the gas consumption in winter for all buildings in a given area. The mobile phone will provide push notification according to the gas usage. Suppose that the command is: *notify me and display all the buildings below the altitude of 600 meters, when the gas consumption in winter is 15% higher than the previous years for the same month*.

This command requires the system to access four types of data. Therefore, to achieve the result, the SPARQL query is developed and accessed four different databases in our system. We can find information about the user's notification configuration from an XML file. We can get the information of building, geography, and weather, from Triplestore. The detail of gas consumption comes from MongoDB, and the monthly aggregate energy data is from MySQL. Figure 2 shows the relationship between the request and the concerning databases.

Different from using a specific query language for each corresponding database, only semantic RDF query language SPARQL is used in our case study. The advantages of this method are more efficiency and rapid data management. In addition, it does not change the original data sources from

the local databases. The following SPARQL scripts are used to get the results:

```
PREFIX db_vocab:
<http://d2r_server/resource/vocab/>
prefix musyopOnto:
<http://www.websematique.ch/voc/musyop#>
prefix dbp-ont:
<http://dbpedia.org/ontology/>
prefix xsd:
<http://www.w3.org/2001/XMLSchema#>

select ?managerID  ?buildingID
?buildingLabel ?readingID ?cons1 ?cons2
?consDiff ?city ?elevation
{
SERVICE <http://d2r_server/sparql> {
?reading a db_vocab:readings;
db_vocab:reading_affectation "Gaz";
db_vocab:reading_building_id ?buildingID
;
db_vocab:reading_id ?readingID.
?monthlyData2011
db_vocab:monthly_data_reading ?reading;
db_vocab:monthly_data_timestamp "2011-
01-01T00:00:00"^^xsd:dateTime;
db_vocab:monthly_data_consumption
?cons1.
?monthlyData2012
db_vocab:monthly_data_reading ?reading;
db_vocab:monthly_data_timestamp "2012-
01-01T00:00:00"^^xsd:dateTime;
db_vocab:monthly_data_consumption
?cons2.
?buildingUser rdf:type
db_vocab:building_user;
db_vocab:building_user_building_id
?buildingID;
db_vocab:building_user_utilisateur_id
?managerID;
}
?building a musyopOnto:Building;
musyopOnto:hasID ?buildingID ;
rdfs:label ?buildingLabel;
musyopOnto:locatedIn ?city.
?city dbp-ont:elevation ?elevation.
Filter(?elevation < ?paramElevationMax)
BIND((xsd:double(?cons2)/xsd:double(?con
s1)) as ?consDiff).
Filter(?consDiff > ?paramConsDiffMax)
}
ORDER BY ?buildingID ?managerID
limit 100
VALUES (?managerID) { MANAGERID_VALUES }
```

Finally, results are returned to the user, including a short description of the issue, the name of the building, date and time. These results can be displayed on the user's mobile device's screen and users also can be notified by SMS and email.

5.3 Design and Development of Mobile Application

We implemented a mobile application according to the case study discussed earlier. The overall goal in designing this application is to examine the feasibility of our approach and to test the performance of our heterogeneous distributed database system with real-time energy data.

This mobile application is developed in HTML5 and PHP. It includes a cross-platform framework called jQuery Mobile [47]. Instead of writing unique applications for each mobile device's operating system, such as iOS, Android, or Windows Phone, the jQuery mobile framework allows us to design a single highly-branded responsive web application that will work on all popular smartphones. The XML files store all settings for the user's configuration. By using the PHP and Java Bridge library [48], we can facilitate the process to connect the online web services to get the final query results from our server.

Our application contains five main functions: (1) configure notification settings, (2) set the active alarms, (3) display all acquitted alarms, (4) manually send a new alarm, and (5) check new alarms from our servers.

The user's notification configuration uses the XML files to store user's settings. As shown on the first interface in Figure 3, the user can select one or more notifications according to his or her preferences. These notifications correspond to the different situations. Once the user receives a notification, he or she could acquit the alarm if the problem has been solved (refer to the second interface in Figure 3). Therefore, all users relevant to this alarm will receive an acquit notification. Moreover, the user can find all the acquitted alarms with detailed information, including the date and time of alarm creation and acquisition, as well as the name of user who acquitted the alarm from a history list (refer to the third interface in

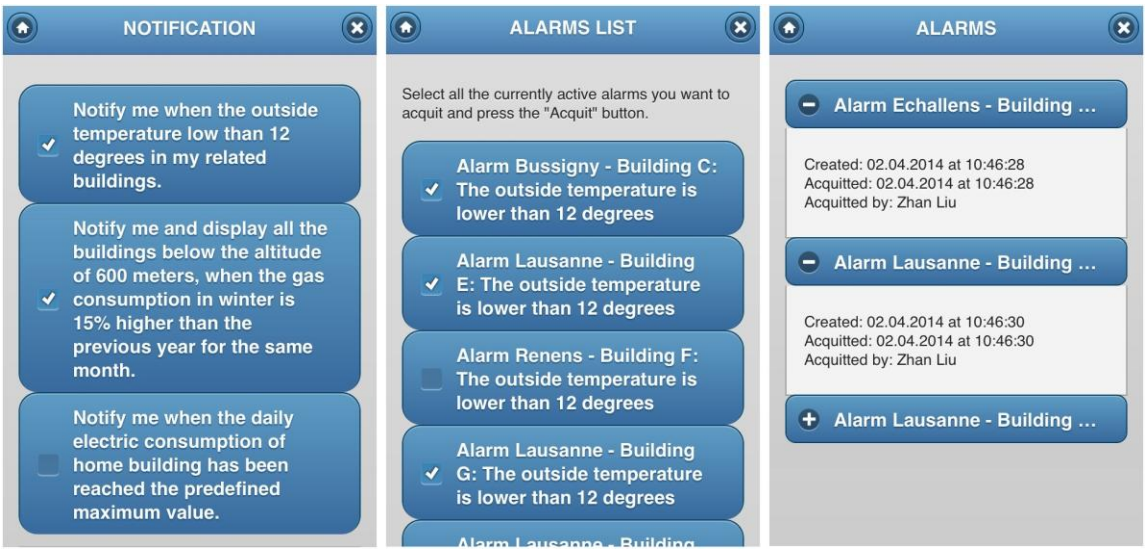

Figure 3. Interface of notifications configuration, active alarms, and acquitted alarms

Figure 3).

There are two channels to send new alarms to users. The first is done manually by the energy consumption manager (refer to the first interface in Figure 4). The manager can describe a new alarm and select one or several users to send. Different users have different levels of access for our application. For example, the new alarm setting can be done only by the energy manager. The doorkeeper of a building, on the other hand, can only receive alarms. The second channel is automatically done by our remote servers with corresponding users' configurations (refer to the second interface in Figure 4). This function would check all new alarms from four different distributed databases through the online web services. The user will then receive the notifications in forms of SMS and email, as described on the third interface in Figure 4. In addition, the user could click the link on the notification message to access the mobile application on the phone with the username and password.

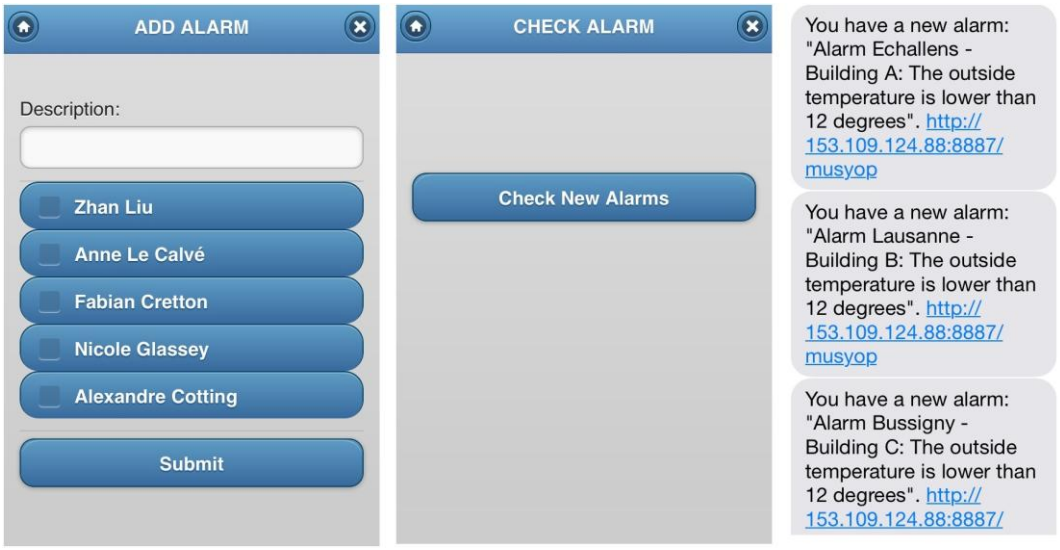

Figure 4. Interface of send new alarm, check new alarms, and acquitted alarms

6 EVALUATION AND DISCUSSION

A first evaluation has been done with a small group of academic and industry experts of energy and database management to whom the architecture has been presented. The structure and methodologies of architecture implementation have been accepted. Feedback in general is quite positive, although some suggestions about the design of user interfaces on mobile phone devices are offered.

A second evaluation of the mobile application related case study has been done with a group of mobile users from different functions, including an energy manager, technicians, computer developers, and building concierges. We conducted this usability test of our mobile application by using the real energy database based on a closed system in a private energy data management enterprise in Switzerland. This evaluation would not only test the feasibility of our whole system, but more importantly, it also would perform a benchmark with existing solutions, including the SPARQL query and storage system performance. In our study, the energy manager was first asked to access the mobile application with a browser on his mobile phone. He then checked new alarms from different remote servers. After that, all push notifications were sent to the mobile devices of related building concierges and technicians. Once the energy problem was resolved by any user, an acquitted confirmation alarm could be sent by this user to other users, informing them that the problem was solved.

The results revealed that most users claimed they liked this mobile application, and they found it very useful compare to existing solutions, such as computer and telephone call. They expressed their intention to use the application for their daily work in the future, because it was more efficient and ease to use. The energy manager and computer developers clearly expressed that the performance of the new heterogeneous distributed database system was better than their existing energy data management solution (e.g., depends solely on relational database).

7 CONCLUSION AND FUTURE RESEARCH

This paper described the architecture of a heterogeneous distributed database system. Contrary to other studies, we proposed a mediator server, a middleware that contains a global schema throughout the network and is used to support and coordinate the distributed transaction management. Based on this mediator, we showed how to query data among four widely used data sources, including relational database, Triplestore, NoSQL database, and XML. With this approach, the system can integrate any kind of component database and it does not require any changes to local databases. We also proposed an approach for query optimization based on our architecture. Moreover, we implemented our solution in a real example in a Swiss energy company and evaluated its performance. We found positive feedback from key person for the distributed data system based on our new architecture. Furthermore, we created and developed a mobile application in HTML5 and PHP based on our solution. This application can successfully provide push notifications about the energy consumption for corresponding users.

In the near future, we plan to experiment to further enhance the flexibility and optimize the query to speedily retrieve data. We also intend to optimize four RDF wrappers in our architecture to improve performance of the whole system.

8 ACKNOWLEDGEMENTS

Financial support for this study was provided by the University of Applied Sciences and Arts Western Switzerland (HES-SO) under grant number 34930, the name of the project is MUSYOP. The authors thank professors Alexandre Cotting and Fabrice Chapuis for their works on implementing the distributed system. We also thank professor Arnaud Zufferey and Frédéric Revaz for their suggestions on the case study of energy data management.

9 REFERENCES

1. Wu, B., Li, J., Liu, H., Zhang, Z., Zhou, Y., and Zhao, N.: Energy Information Integration based on EMS in Paper Mill. Applied Energy 93, 488--495 (2012).
2. Ueno, T., Sano, F., Saeki, O, and Tsuji, K.: Effectivenss of an energy-consumption information system on energy saving in residential houses based on monitored data. Applied Energy 83(2), 166--183 (2006).
3. Smith, J. M., Bernstein, P. A., Dayal, U., Goodman, N., Landers, T., Lin, K. W. T., and Wong, E.: Mutibase - Integrating Heterogeneous Distributed Database Systems. In: Proc. AFIPS of the National Computer Conference, pp.487--499, Chicago, USA (1981).
4. 10gen, Inc.: MongoDB documentation (2012).
5. Karatzas, K., Papadopoulos, A., Moussiopoulos, N., Kalognomou, E. A., and Bassoukos, A.: Development of a Hierarchical System for the Tele-transmission of Environmental and Energy Data. Telematics and Informatics 17, 239--249 (2000).
6. Giacomazzi, F., Panella, C., Pernici, B. and Sansoni, M.: Information systems integration in mergers and acquisitions: A normative model. Information and Management 32(6), 289--302 (1997).
7. Chaudhuri, S., and Dayal, U.: An Overview of Data Warehousing and OLAP Technology. ACM SIGMOD Record 26(1), 65--74 (1999).
8. Sheth, A. P., and Larson, J. A.: Federated Database Systems for managing Distributed Heterogeneous, and Autonomous Databases. ACM computing Surveys 22(3), 183--236 (1990).
9. Haas, L. M, and Soffer A.: New Challenges in Information Integration. In: DaWak 2009, Data Warehousing and Knowledge Discovery, Pedersen, T., Mohania, M., Tjoa, A. (eds.) pp. 1--8. Springer, Heidelberg (2009).
10. Berners-Lee, T., James, H., and Ora L.: The Semantic Web. Scientific American Magazine (2001).
11. Theocharis, S. A., and Tsihrintzis, G. A.: Semantic Web Technologies in e-Government. World Academy of Science, Engineering and Technology 64, 1237--1244 (2012).
12. Wang, J., Miao, Z., Zhang, Y., and Zhou, B.: Querying Heterogeneous Relational Database Using SPARQL. In: Proc. Eighth IEEE/ACIS International Conference on Computer and Information Science, pp.475--480 (2009).
13. Chen, Y., Mu, X., Zhang, J., and Lu, Z.: Development of monitoring system of building energy consumption. In: Proc. International Forum on Computer Science-Technology and Applications, vol. 2, pp. 363--366, IEEE Computer Society (2009).
14. Haberl, J., Sparks, R., and Culp, C.: Exploring new techniques for displaying complex building energy consumption data. Energy and Building 24, 27--38 (1996).
15. Qiu, D., Zhang, D.: The research and realization of energy management system in iron and steel. In: Proc. International Conference on Information Networking and Automation, vol. 1, pp. 448--451 (2010).
16. European Commission: Energy-efficient buildings: Challenges ahead (2013), http://ec.europa.eu/research/industrial_technologies/eeb-challenges-ahead_en.html
17. Pargfrieder, J., Jorgl, H.P.: An integrated control system for optimizing the energy consumption and user comfort in buildings. In: Proc. of Computer Aided Control System Design, pp.127--132 (2002).
18. Vidrih, S., Umberger, M., Humar, I.: Optimization of electric energy consumption operating costs in cooling technique by establishing a remote central control system, exploiting thermal capacity and load shifting. In In: Proc. of Eurocon, pp.1479--1484 (2013).
19. Mazzocchi, S., Garland, S., and Lee, R.: Simile: Practical metadata for the semantic web. XML.com (2006).
20. Josifovski, V., Schwarz, P., Haas, L., and Lin. E.: Garlic: A New Flavor of Federated Query Processing for DB2. In: Proc. the 2002 ACM SIGMOD International Conference on Management of Data, pp. 524--532, Madison, Wisconsin (2002).
21. Schenk, S., Saathoff, C., Staab, S., and Scherp, A.: SemaPlorer – Interactive Semantic Exploration of Data and Media based on a Federated Cloud Infrastructure. Journal on Web Semantics: Science, Services and Agents on the World Wide Web 7(4), 298--304 (2009).
22. Arenas, M., and Pérez, J.: Querying Semantic Web data with SPARQL. In: Proc. Principles of Database Systems, ACM, pp. 305--316 (2011).
23. Langegger, A., Wöß, W., and Blöchl, M.: A Semantic Web Middleware for Virtual Data Integration on the Web. In: Proc. 5th European semantic web conference on the semantic web: research and applications, pp. 493--507, Springer Berlin, Heidelberg (2008).
24. Chen H., Wang, Y., Wang, H., Mao, Y., Tang, J., Zhou, C., Yin, A., and Wu, Z.: Towards a semantic web of relational databases: a practical semantic toolkit and an in-use case from traditional chinese medicine. In: Proc. 4th International Semantic Web Conference (ISWC), LNCS, pp. 750--763, Springer-Verlag, Athens, USA (2006).
25. de Laborda, C. P., and Conrad, S.: Bringing Relational Data into the SemanticWeb using SPARQL and Relational OWL. In: Proc. 22nd International Conference on Data Engineering Workshops (ICDEW'06), pp. 55 (2006).
26. Gorlitz, O., and Staab, S.: Federated Data Management and Query Optimization for Linked Open Data. New Directions in Web Data Management, Springer, pp. 109--137 (2011).
27. Hartig, O., and Heese, R.: The sparql query graph model for query optimization. In: Proc. 4th European Semantic Web Conference (ESWC), pp. 564--578 (2007).
28. Kossmann, D.: The State of the Art in Distributed Query Processing. ACM Computing Surveys 32(4), 422--469 (2000).

29. Schmidt, M., Meier, M., and Lausen, G.: Foundations of SPARQL query optimization. In: Proc. 12th International Conference on Database Theory, pp. 4--33 (2008).
30. Yeh, D. Y., Lee, M. C., and Wang, T. I.: Mobile Agents for Distributed Transactions of a Distributed Heterogeneous Database System. In: Proc. 13th International Conference on Database and Expert Systems Applications (DEXA 2002), pp. 403--412, Aix-en-Provence, France (2002).
31. Bizer, C., and Seaborne, A.: D2rq: Treating non-rdf databases as virtual rdf graphs. In: Proc. 3rd International Semantic Web Conference (ISWC2004 posters) (2004).
32. Stavrakantonakis, I., Tsinaraki, C., Bikakis, N., Gioldasis, N., and Christodoulakis, S.: Sparql2xquery 2.0: Supporting semantic-based queries over xml data. In: Proc. 5th Semantic Media Adaptation and Personalization (SMAP), pp. 76--84, IEEE (2010).
33. SPARQLverse (2014), http://sparqlvillage.org/documentation/
34. Schwarte, A., Haase, P., Hose, K., Schenkel, R., and Schmidt, M.: FedX: Optimization Techniques for Federated Query Processing on Linked Data. In: Proc. 10th International Semantic Web Conference, pp. 481--486, Bonn, Germany (2011).
35. Auer, S., Dietzold, S., Lehmann, J., Hellmann, S., and Aumueller, D.: Triplify – Light-Weight Linked Data Publication from Relational Databases. In: Proc. 18th World Wide Web Conference, pp. 621--630, Madrid, Spain (2009).
36. Rodriguez, J.B., and Gomez-Perez, A.: Upgrading relational legacy data to the semantic web. In: Proc. 15th international conference on World Wide Web, pp. 1069--1070, Edinburgh, Scotland (2006).
37. Cerbah, F.: Learning highly structured semantic repositories from relational databases: the RDBToOnto tool. In: Proc. 5th European semantic web conference on The semantic web: research and applications, pp. 777--781, Tenerife, Canary Islands, Spain (2008).
38. Bray, T., Paoli, J., Sperberg-McQueen, C. M., Maler, E., and Yergeau, F.: Extensible Markup Language (XML) 1.0 (Fifth Edition). World Wide Web Consortium, Recommendation REC-xml-20081126 (2008).
39. Goldman, R., McHugh, J., and Widom, J.: From semistructured data to XML: Migrating the Lore data model and query language. In: Proc. ACM SIGMOD WebDB Workshop, pp. 25--30 (1999).
40. Suciu, D.: Semistructured data and XML. Information organization and databases, pp. 9--30, Springer US (2000).
41. Droop, M., Flarer, M, , Groppe, J., Groppe, S., Linnemann, V., Pinggera, J., Santner, F., Schier, M., Schopf, F., Staffler, H. and Zugal, S.: Embedding XPATH Queries into SPARQL Queries. In: Proc. 10th International Conference on Enterprise Information Systems (ICEIS), pp. 5--14 (2008).
42. Akhtar, W., Kopecky, J., Krennwallner, T. and Polleres, A.: XSPARQL: Traveling be-tween the XML and RDF Worlds and Avoiding the XSLT Pilgrimage. In: Proc. 5th European Semantic Web Conference, ESWC 2008, pp. 432--447, Tenerife, Canary Islands, Spain (2008).
43. McBride, B.: Jena: A semantic web toolkit. IEEE Internet computing 6(6), 55--59 (2002).
44. Broekstra, J., Kampman, A., and Van Harmelen, F.: Sesame: A generic architecture for storing and querying rdf and rdf schema. In: Proc. the Semantic Web-ISWC 2002, pp. 54--68, Springer Berlin Heidelberg (2002).
45. Bizer, C., and Schultz, A.: The Berlin SPARQL Benchmark. International Journal on Semantic Web & Information Systems 5(2), 1--24 (2009).
46. SPARQL 1.1 Federated Query (2013): http://www.w3.org/TR/sparql11-federated-query/
47. Charland, A., and Leroux, B.: Mobile application development: web vs. native. Communications of the ACM 54(5), 49--53 (2011).
48. PHP and Java bridge, http://php-java-bridge.sourceforge.net/pjb/index.php