A Semantic Data Federation Engine: Design, Implementation & Applications in Educational Information Management

by

Mathew Sam Cherian

B.S. Electrical Engineering, Johns Hopkins University (2006)

Submitted to the Engineering Systems Division

and

Department of Electrical Engineering and Computer Science
in partial fulfillment of the requirements for the degrees of

Master of Science in Technology and Policy

and

Master of Science in Computer Science and Engineering

at the

MASSACHUSETTS INSTITUTE OF TECHNOLOGY

February 2011

Author . . . .
Engineering Systems Division
Department of Electrical Engineering and Computer Science
January 26, 2011

Certified by . . . .
Lalana Kagal
Research Scientist, Computer Science and Artificial Intelligence Laboratory
Thesis Supervisor

Accepted by . . . .
Professor Dava Newman
Director, Technology and Policy Program

Accepted by . . . .
Professor Terry P. Orlando
Chair, Committee on Graduate Students
Department of Electrical Engineering and Computer Science

A Semantic Data Federation Engine: Design, Implementation & Applications in Educational Information Management

by

Mathew Sam Cherian

Submitted to the Engineering Systems Division
and
Department of Electrical Engineering and Computer Science
on January 26, 2011, in partial fulfillment of the
requirements for the degrees of
Master of Science in Technology and Policy
and
Master of Science in Computer Science and Engineering

Abstract

With the advent of the World Wide Web, the amount of digital information in the world has increased exponentially. The ability to organize this deluge of data, retrieve it, and combine it with other data would bring numerous benefits to organizations that rely on the analysis of this data for their operations. The Semantic Web encompasses various technologies that support better information organization and access.

This thesis proposes a data federation engine that facilitates integration of data across distributed Semantic Web data sources while maintaining appropriate access policies. After discussing existing literature in the field, the design and implementation of the system including its capabilities and limitations are thoroughly described. Moreover, a possible application of the system at the Massachusetts Department of Education is explored in detail, including an investigation of the technical and non-technical challenges associated with its adoption at a government agency.

By using the federation engine, users would be able to exploit the expressivity of the Semantic Web by querying for disparate data at a single location without having to know how it is distributed or where it is stored. Among this research's contributions to the fledgling Semantic Web are: an integrated system for executing SPARQL queries; and, an optimizer that faciliates efficient querying by exploiting statistical information about the data sources.

Thesis Supervisor: Lalana Kagal
Title: Research Scientist, Computer Science and Artificial Intelligence Laboratory

Acknowledgments

I would like to thank my advisor Lalana Kagal for all the guidance and support she has given me over the past two years. She has constantly encouraged me to pursue opportunities in research and life.

This thesis benefited greatly from the interactions I had with other members of DIG. Particularly, I am grateful to Hal Abelson for taking the time to listen to my proposal to investigate the applications of my research to educational information management, a topic which is only remotely related to the Decentralized Information Group's core mission, and putting me in touch with officials at the Massachusetts Department of Elementary and Secondary Education; I am indebted to Tim Berners-Lee for the many brief chats we had, which gave me more insights than what I got by reading academic publications.

I had a great experience working with students and staff at the Decentralized Information Group and the World Wide Web Consortium. Specifically, I want to thank Eric Prud'hommeaux for being a C++ guru and telling me more Emacs shortcuts than I can possibly remember; Alexandre Bertails for convincing me to convert and stay with the Ubuntu; and, Oshani Seneviratne for patiently answering the many questions I asked and for always willing to lend a helping hand.

This thesis, and my time at MIT would not have gone as well as it did, if it were not for the constant interactions I have had with the following people: Teena, thank you for always making me feel good about myself in spite of the drags of graduate student life; Tony, thank you for always being there for a chat, even if it was after you had been cutting up cadavers all day; and, Thomas for always knowing the right questions to ask to help me make it through the many difficult times.

I would not be here today without the efforts of my parents, who work tirelessly every day of their lives for their three children. Amma and Acha: Thank you for everything.

Contents

List of Figures

List of Tables

Listings

Chapter 1

Introduction

The amount of digital information in the world has exploded in the past decade. In 2005, about 150 exabytes of data were created - a number that has been estimated to have reached 1200 exabytes in 2010 [34]. Detecting patterns and extracting useful information from this deluge of data can significantly benefit firms, governments, and people by improving efficiencies, and innovating products and practices, among others. Given the size of this data, it is necessary to have appropriate tools that can facilitate the organization and querying of data, which is often contained in disparate locations. The Semantic Web has the potential to make such large scale data manipulations possible.

The term Semantic Web encompasses the broad category of technologies that are aimed at making the World Wide Web more machine-readable. Some of these technologies include Resource Description Framework (RDF) [22], RDF Schema (RDF-S) [8], Web Ontology Language (OWL) [31], the SPARQL Query Language [39]. Though the Semantic Web has led to open data sources that can be queried, extended and re-used easily [35], on-the-fly integration of data from multiple heterogeneous and distributed data sources is still a problem. Furthermore, the ability to perform such seamless integrations while preserving appropriate security and privacy policies has not even been addressed.

Several issues need to be addressed before such environments are developed. For instance, an application developer/user would need to know what data is present in each source and in which format, so that appropriate queries may be formulated to obtain the necessary information. Moreover, if a source's content has been modified, the application developer needs to adjust the query accordingly to ensure proper data retrieval. One possible solution to address these challenges is the application of the federation paradigm for database systems [32].

A federated query involves the concurrent search of multiple distributed data sources. In a federation model, the access to data sources is transparent. Under such a system, a user can submit a single query to the federation engine, which then reformulates the query to subqueries to suit the various data sources, executes the subqueries, and returns the merged result. The lack of a shared model for security and privacy [20], however, impedes the transparency of the federated model because the federation engine is unable to execute requests across many secure domains on the fly. This means that most contemporary federations have static, previously negotiated policies in place, which prevents dynamic integration of data.

In this thesis, I present a novel architecture of a data integration engine that provides secure SPARQL federation. The engine accepts a single SPARQL query from a client, splits it up into subqueries based on the information contained in the service descriptions of endpoints it has in its possession, sends them off to appropriate SPARQL endpoints, and returns the results to the client. Under the proposed architecture, clients who use the federation engine are also required to provide some security credentials, which are used to satisfy the access policies of those endpoints that contain secure data. Such data may include personally identifiable information(PII) of individuals and trade secrets, among others.I also present my implementation of this architecture, which includes provisions for integration with Semantic Web-based policy reasoners[43, 16], but does not currently support complete policy enforcement.

1.1 Thesis Overview

The proposed system for SPARQL federation is organized in this thesis as follows:

Chapter 2 introduces some of the basic technologies that makes query federation possible in a Semantic Web environment. A basic overview of federated querying is provided and the differences between relational and RDF databases are illustrated. Then, some of the recent advances in federation functionality in Semantic Web technologies is explored.

Chapter 3 describes the application of a SPARQL federation system in the context of government data sharing. When organizations see the business case for expressing their data in formats that suit the Semantic Web, they would begin developing data systems that facilitate enterprise level data sharing. The presence of such data sources would accelerate the spread of Semantic Web Technologies on a truly Web scale. This detailed use case highlights the many advantages of adopting SPARQL federation systems for the Massachusetts Department of Education over traditional databases.

Chapter 4 discusses the overall architecture of the system and the design choices that were made during its development. The various components of the system are described and their features and limitations are identified.

Chapter 5 details the implementation of the design. The various software tools that made the design possible are identified and their relevance is described.

Chapter 6 discusses the performance of the system when presented with endpoints and queries with different characteristics. The methodology for the selection of queries is presented and the relevant metrics are illustrated.

Chapter 7 provides a summary of the contributions of this work. The limitations of the system are discussed and future areas for expanding on the work are identified.

Chapter 2

Background & Related Work

In order to design a federation engine for Semantic Web data sources, it is first necessary to compare Semantic Web technologies to their relational counterparts. Moreover, many insights for the engine's design can be found in the existing literature on federated databases. In the following sections, these results of this two pronged investigation is described.

2.1 Relational Databases

Most of the research related to database federation has been conducted on relational databases. This is no surprise because the relational model has been the dominant model since it was first described by by E.F. Codd in [11]. A relational database has a tabular structure, with each column in the table signifying a different attribute. A relation is defined as a set of tuples that contain a set of elements, one each from each attribute. In Tables 2.1, 2.2, and 2.3 are sections of three tables that are part of a relational database for the Olympic games. Table 2.1 contains a list of host cities and their geolocation information. Table 2.2 contains the dates of the events and their mottoes. Table 2.3 has the details of athletes attending the events.

Table 2.1: Example of Relational Database Table: Host Cities

idville	label-en	longitude	latitude
1	Athens	23.7166667	37.966667
2	Beijing	116.400002	39.900002

Table 2.2: Example of Relational Database Table: Event Details

id	idville	year	Opening Ceremony Date	Closing Ceremony Date	Motto
1	1	1896	1896/04/06	1896/04/15	NULL
2	2	2008	2008/08/08	2008/08/24	One World, One Dream

2.1.1 Limitations of the Relational Model

The relational model has a number of shortcomings. First, from Tables 2.1, 2.2, and 2.3, it is evident that there is a clear separation between the data structure (the attributes) and the data itself. This means that the relationship between the data and the attributes are not persistent. Second, the database has a rigid structure, which means that it is necessary to have NULL values for those attributes that do not exist in a relationship. For instance, the 1896 Olympic games did not have a motto but it is necessary to specify that fact explicitly. Third, the relationships between the tables are not explicitly stated and have to be inferred. To gather all the non-athlete information on a particular Olympic games, it is necessary to do a join on Tables 1 & 2 using *idville*. Fourth, all data and attribute identifiers are local to a given database. This means that they lack meaning outside the database and therefore cannot easily used for data manipulations across different data sources [7].

2.2 Database Federation

The research on federated database systems dates back to the 1980s when McLeod and Heimbigner first detailed a federated approach to managing decentralized database

Table 2.3: Example of Relational Database Table: Athlete Details

Athlete ID	label-en	Birth Date	Death Date
2	Beijing	116.400002	39.900002

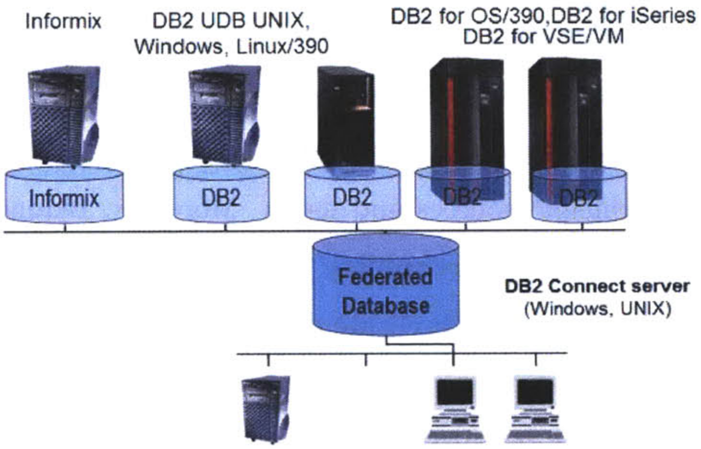


Figure 2-1: A Database Federation [33]

systems. They defined a decentralized database as a collection of structured information that may be logically and/or physically distributed. Logical decentralization is the division into components for allowing separate access, and physical decentralization is about the allocation of data to different nodes. A federated system consists of a number of logical components, such as endpoints, related, but independent, and each with its own component schema, defined by a component Database Administrator (DBA). A schema describes the structure of a database in formal language. The principal goal of each component is to satisfy the needs of its most frequent and important users, which are usually the local ones. These disparate components are held together by a federal schema, which contains a federal controller and is administered by a federal DBA. The federal DBA resolves conflicts between components and defines federal schema(s), relate them to component schemas, and specifies each components interface.

There are many options available for logical distribution - a single, global federal schema, a separate federal schema for each pair of components, associating a federal schema with each component, or a hierarchy of federal schemas. Given a federated setup, there is a range of possible views that a given component user can have. At one end, the federal and local schemas are so well integrated that the user cannot tell which data is being accessed. In this setup, the user could be oblivious to potentially expensive non-local references. On the other end, the federal schema is separate from the local one and therefore the user would have to address each one separately.

With physical distribution, better performance would result as a result of placing the data close to principal sources and users. Moreover, redundantly storing data would provide reliability and survivability. In such implementations, the complexity that exists from the presence of duplicate data as well as that which results from the combination of logical and physical decentralization must be addressed.

The federal controller (or engine) is the entity that is charge of the federation. The controller needs to perform several steps for each request from a client.

1. The request is checked for validity against the federal schema and access rights are verified.

2. The request is decomposed into several subrequests, each of which can be satisfied by a single component.

3. Each subrequest, if applicable, is translated from the federal schema to the target component schema.

4. The subrequests are sent to the corresponding components.

5. The results from each component is collected and, if applicable, translated from the component schema to the federal schema.

6. The results are combined and returned to the client.

There are three approaches to federal controller placement - as a special node on the network, co-located on a node with another component, or parts of it may be distributed across different components. The component controllers should have three important features: allow local users and the federal controller concurrent access to data, communicate results back to federal controller, and recognize locally-issued requests for which the federal schema is needed and forward those to the federal controller [32]. In this thesis work, the objective was to create an independent, standalone federation engine that could be interfaced with any combination of components. As a result, the first approach was chosen.

2.3 Semantic Web

The Semantic Web is touted and increasingly accepted as the next generation of the Web. The inventor of the the World Wide Web described the Semantic Web as an extension of the current Web in which information has well-defined meaning that is machine readable [5]. As such, Semantic Web technologies provide a framework for sharing information between decentralized sources using data interpretation, integration, and management capabilities. In such an environment, there would be significant reductions in the hours spent on decision-making, organizing, and filtering useful knowledge from the human-only-readable, large data repository that is the Web today.

A number of features enable the structure and machine readability of the Semantic Web. Unique Resource Identifiers (URI) enable unique meaning, the Resource Description Framework (RDF) facilitate the meta data layer, and SPARQL - a query language for RDF - permits the efficient and descriptive querying of Semantic Web sources.

2.3.1 Uniform Resource Identifier (URI)

URIs can be Uniform Resource Locators (URLs), such as web addresses, or Uniform Resource Names (a unique name within a given namespace) [42]. A URL, such as *http://dig.csail.mit.edu*, identifies the homepage of the Decentralized Information Group and also indicates the location of that information. A URN, on the other hand identifies a resource, so that there is no ambiguity in its reference. It does not necessarily give any information on the resource's location. For example, *dig:source_descriptions*, identifies the namespace for the source description ontology for the system described in this thesis. URIs ensure that a given name will always

```
@prefix foaf:  <http://xmlns.com/foaf/0.1/> .

_:a  foaf:name   "Johnny Lee Outlaw" .
_:a  foaf:mbox   <mailto:jlow@example.com> .
_:b  foaf:name   "Peter Goodguy" .
_:b  foaf:mbox   <mailto:peter@example.org> .
_:c  foaf:mbox   <mailto:carol@example.org> .
```

Listing 2.1: Sample RDF

correspond to the same resource across domains such that disparate information may be brought together for integration [45].

2.3.2 Resource Description Framework (RDF)

RDF is a directed, labeled graph data format for knowledge representation on the Web [22]. RDF is used to describe the attributes of resources and establish relationships between different resources. Relationships are established via subject-predicate-object triples, in which the predicate identifies a relationship between two resources - the subject and the object. The various components of a triple would ideally made up of URIs so that the triples maybe easily processed and manipulated. Listing 2.1 shows part of an RDF Graph, expressed as a set of triples. These triples together contain the name and/or email information for three individuals.

2.3.3 SPARQL Protocol and RDF Query Language (SPARQL)

SPARQL is an RDF query language [39]. It can be used to express queries to a single data source or across multiple data sources. SPARQL is capable of querying required and optional graph patterns along with their conjunctions and disjunctions. The results of a SPARQL query may be returned as a result set or RDF graphs. The expressive power of SPARQL is perhaps best demonstrated by an example. Suppose that we have are interested in querying the data in Listing 2.1. We could create a query that would pull pairs of names and e-mail addresses from this data set. In SPARQL, this query would have the syntax shown in Listing 2.2.

```
PREFIX foaf:    <http://xmlns.com/foaf/0.1/>
SELECT ?name ?mbox
WHERE
  { ?x foaf:name ?name .
    ?x foaf:mbox ?mbox }
```

Listing 2.2: Sample SPARQL Query

The real power of SPARQL is the fact that it is already conveniently encoded in an RDF language. When a query is executed, an attempt is made to match the Basic Graph Pattern (BGP), i.e. the combination of triples, in a SPARQL query to any part of the RDF Graph it is trying to query.

The result of this particular query would be names and mboxes of individuals for whom both pieces of information is in the RDF Graph. Hence, Carol's information would not be returned as she does not have a *foaf:name* attribute associated with her in the Graph.

2.3.4 Advantages of RDF Data Sources

A juxtaposition of the features of the Semantic Web and the aforementioned limitations of the relational mode illustrates how the RDF model makes up for these deficiencies. The use of URIs mean that the reference to a data element has meaning across multiple domains. Such persistence makes data integration, interpretation, and management across different data sources seamless. Moreover, the graph structure of the data is very flexible. The DBA is free to choose any shape for the data she is in charge of. Also, the use of triples ensure that the relationships between database elements are explicit and easy to discern with appropriate SPARQL queries.

2.4 Related Work

Much of the research on database federation has focused on relational databases. Some of this also includes work on secure relational federations. For example, HER-

MES (HEterogeneous Reasoning and MEdiator System) developed at the University of Maryland was a platform that was developed to design and implement such systems [9]. There has been little work done on such comprehensive environments for Semantic Web, especially secure ones, due to the relative newness of such technologies. However, marked progress has been made in the development of various components that make up a federation engine for the Semantic Web.

Some work has been done on the semantic integration of relational data using SPARQL [44]. In this particular example, a mediator-wrapper architecture is used for data integration, which is divided into three sub-parts - mediated schema, query language, and query rewriting algorithms. Ontology is used for mediated schema and conjunctive queries are rewritten using Views through the implementation of the MiniCon algorithm. However, since MiniCon uses Datalog, the query processor translates SPARQL queries to Datalog to perform all steps involved in data integration optimization, query rewriting, and query optimization. Moreover, this system only interacts with relational databases and not SPARQL endpoints. Without SPARQL, most queries will lack expressivity and the full potential of the Semantic Web will remain untapped.

Virtual integration of distributed, heterogeneous data sources based on the Semantic Web technology to facilitate scientific collaboration was described in [30]. The authors propose a middleware based on a mediator-wrapper architecture that can map local data models and schemas to several global domain ontologies. Clients submit SPARQL queries corresponding to the global domain ontologies to the mediator. The mediator, which holds information on registered data sources and global domain ontologies, generates the query plan using an iterative dynamic programming algorithm. The wrappers help overcome the model and schema heterogeneity by providing specific access methods. While this system is promising, little performance data of the optimizer and the engine is available that sheds light on the effectiveness of this approach.

Kolas has developed a query rewriting algorithm that attempts to apply lessons learned from relational information integration systems to Semantic Web technologies [28]. However, the focus of this paper is only on one aspect of query rewriting - mapping between local and target schemas. Two approaches are presented and one is chosen (Global-as-View) based only on a number of factors associated with local to global mapping - independent domain ontology, mutually independent data source to domain ontology mappings, extracting all meaning from source to domain ontology, etc. Very little focus is placed on the rewriting process once the mappings are performed.

The OptArq system has implemented a SPARQL optimizer based on triple pattern selectivity. The optimizer aims to reduce the size of the intermediate result sets of triple patterns. An overall cost function, which is a function of the costs of subject, predicate, and object, is used to rank triple patterns. The elementary cost functions are calculated based on statistical data of endpoints that the OptARQ system has cached [6]. Although the statistical models are basic and imprecise, this system provides a good framework for optimizing SPARQL queries for use in federated Semantic Web environments.

The only known federated system that uses SPARQL to query RDF data sources that we are aware of is the DARQ. It is a full-fledged engine that performs query parsing, query planning, query optimization, and query execution. Planning and optimization are done using the source descriptions that the system has on file for those systems that have registered with it ahead of time. DARQ adapts much of the research on federation of relational databases to perform SPARQL queries on RDF sources. However, DARQ only operates on open data sources and does not offer any support for secure SPARQL federation [40]. Secure data sources are often necessary in scientific, business, and socio-political fields for economic and legal reasons, as we will see in chapter 3.

Chapter 3

Public Education in Massachusetts: An Ideal Application

With the emergence of the Web, technologists, policymakers, and citizens have praised the ability of information systems to transform governments. Various entities have enumerated the benefits of e-government systems. These fall into the broad categories of benefits to citizens, businesses, and government agencies.

As e-government systems promote easier access to government information, citizens would have a better customer service experience when interacting with government agencies. The data.gov and data.gov.uk initiatives by the government of the United States and the United Kingdom, respectively, are prime examples of services providing easier access to government information. Businesses would save on operating costs as paper-based processes are eliminated and government information is accessed more efficiently. The agencies that employ the systems would also see cost savings and increased efficiencies as well as increase in morale among agency employees, which feeds back from better customer service [38, 19].

To achieve the full benefits of e-government systems, it is necessary to share information between different government systems. Public policy issues are often spread across different agencies and it is the case that different entities have to marshall their resources to solve a particular problem. For example, leukemia rates among the graduates of River Valley High School in Marion, Ohio increased 122 percent between

1966 and 1992. However, it took almost 30 years for the community to establish a hypothesis - leaking of cancer causing chemicals from a World War II bomb-making plant which used to be housed at the school site - and collect enough evidence to prove it. Had there been systems that allowed interoperability and sharing among Department of Defense (DoD), Census Bureau, and epidemiological information systems, the correlation process would have been easier and taken much less time, and possibly resulted in saving lives [24].

3.1 Interoperability and Data Sharing

In the last 15 years, much research has been done in the area of interoperability between disparate government information systems to enable better collaboration between agencies [24, 15, 1]. Figure 3-1 depicts the theoretical model of inter-agency information sharing that Dawes laid out in [15]. The impetus for these efforts during the last decade was the terrorist attacks of September 11, 2001 and the belief that the tragedy might have been avoided had there been better information exchange between various law enforcement agencies. The *Accountable Information Usage in Fusion Center Information Sharing Environments* project at the Decentralized Information Group (DIG) at MIT-CSAIL attempts to build policy tools that enable seamless information flow between Federal, State, and Local law enforcement agencies, while assuring that such sharing adheres to privacy and securing policies [21]. The goal of this thesis work is not to develop policy tools; rather to use those tools in combination with others to develop a well integrated data federation environment. Data Sharing and Interoperability, moreover, has applications and benefits that extend far beyond the national security and law enforcement realms.

3.1.1 Benefits

Interoperability and the information sharing that results from it has the potential to make agencies more effective, efficient and responsive. Although, many federal agencies were created to solve public policy challenges in their purview - education,

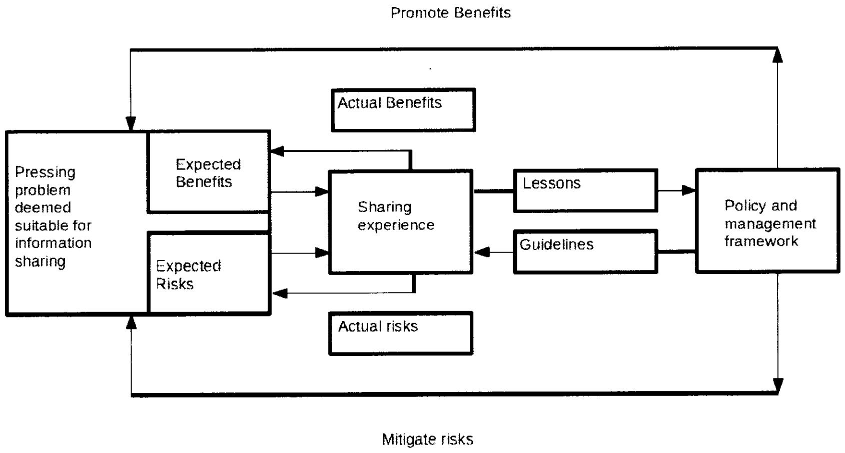


Figure 3-1: Theoretical Model of Inter-agency Information Sharing

environmental protection, etc., policymakers increasingly understand that many of the social and regulatory problems do not conform to the jurisdictional boundaries of a particular organization. These problems, and therefore effective government, require integrated policy approaches that makes use of information, knowledge, and technology from a variety of agencies and divisions [24].

Once information is aggregated in electronic sources, data manipulation and duplication, which are integral to find correlations and causalities, becomes much easier. Such sources have the ability to reduce the paperwork burden on both governmental and non-governmental actors that interact with the state [24]. In the same vein, transaction costs are reduced, thereby bringing down overall costs and possibly increasing participation [15].

Finally, the availability of disparate information at the fingertips of public servants means that governments would act faster to identify problems and respond to them. Moreover, such increases in efficiencies could establish a cadre of entrepreneurial bureaucrats, who are less bound by rigid policies and have the flexibility to tailor the processes to reach goals, the completion of which they are held responsible for [24].

3.1.2 Challenges

While the aforementioned benefits makes a strong case for interoperability, a number of challenges have to be addressed for the full promise of data sharing to be realized. Andersen and Dawes in [1] identify these challenges into the broad categories of political, organizational, economic, and technical issues.

Political obstacles include privacy, ambiguity about statutory authority, and the openness to public scrutiny. Privacy includes the obvious threat of data aggregation by Big Brother governments [24]. In addition to establishing legal frameworks that protect citizens' privacy rights, citizens must also have faith in the government for interoperable systems to become politically viable [23]. Moreover, federal and state agencies are bound by the statutes that authorized their establishment. Under these tight constraints, an important question for an agency is if and under what conditions it can share information with its sister agencies. Those agencies that are hesitant to share would require explicit regulatory direction to divulge information that they possess [15]. Once again, appropriate legal frameworks need to be developed for sharing to become ubiquitous. However, even with appropriate legal frameworks, agencies that wish to withhold certain information may choose to interpret the legislation differently from a partner agency. Lastly, the data that is made easily available through interoperability might cause the public to increase their scrutiny on the agencies that shared the information. This outcome is a strong disincentive for sharing.

In the organizational realm, an agency must trust the veracity of the information it gets from another. For instance, an agency might not be due diligent in ensuring the accuracy of information it collects, if it is not critical to its core mission. However, for data sharing to be a success, each interoperating agency must have a high degree of confidence in the information that it did not directly solicit. Moreover, the fledgling state of the development of interoperable systems means that agencies lack the expertise to share the information they have and/or are unaware of the opportunities that exist for sharing. To complicate matters further, the agencies often lack the financial resources to secure interoperable systems, which are often state-of-the-art and have high costs. Moreover, cash-strapped agencies often use low-bid procurement methods for information systems, which fail to take into account the life-time impact of such systems. These are significant economic obstacles to interoperability [24].

Last, but not least, there are a number of technical challenges. Prime among them are issues of hardware and software incompatibilities. While there may be workarounds, the cost of them might be prohibitive. Moreover, when contractors are used for system development, intellectual property rights of the contractor might prevent the contracting agency from not utilizing all the features of the system. Finally, a key technical barrier to interoperability is the difference in data standards. It would be difficult to connect disparate datasets if agencies have different data definitions. While these definitions could be changed for the purpose of integrating diverse systems, the financial and legal barriers of doing so might be too high [24]. Usage of RDF and other Semantic Web technologies described in section 2.3 could aid in integration while allowing each agency to customize its data structures for local needs.

Section 3.3 lists some of the ways in which these challenges may be mitigated in an environment that uses SPARQL federation for data sharing.

3.2 Public Education in Massachusetts

Education is an important public sector service where interoperability can have tremendous benefits. In Massachusetts, the Department of Elementary and Secondary Education (DESE)[1] is responsible for the education of the approximately 550,000 children in the state's public schools, which are located in 391 school districts. Its mission is *To improve the quality of the public education system so that students are adequately prepared for higher education, rewarding employment, continued education, and responsible citizenship.* It has as one of its six primary goals the

[1] Note: Massachusetts has had many reorganizations of the state level education administration in the last decade. In this thesis, the term DESE is used to identify the Department of Elementary and Secondary Education as well as its predecessors

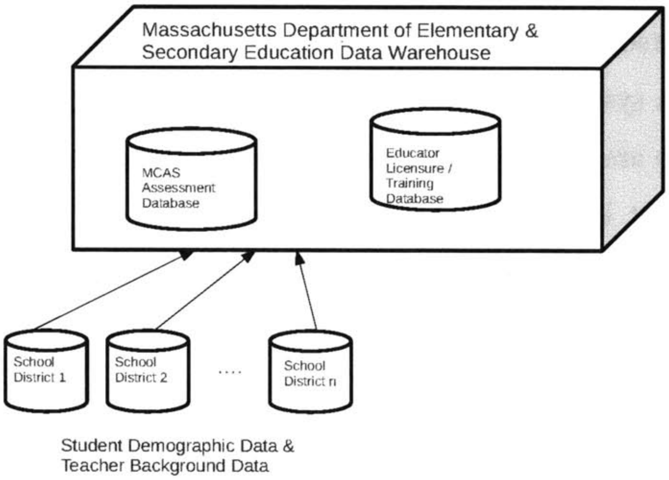

Massachusetts Department of Elementary & Secondary Education Data Warehouse
MCAS Assessment Database
Educator Licensure / Training Database
School District 1
School District 2
....
School District n
Student Demographic Data & Teacher Background Data

Figure 3-2: Massachusetts DESE Data Sharing for Accountability

provision of timely, useful information to stakeholders [12]. To achieve its mission and goals, it is important for the DESE to track the progress of students as they advance through the grades. Moreover, it is necessary to address the needs of children in early childhood and in the post-secondary years, when they are not in the purview of the DESE. Without such attention, we would lack an active citizenry that sustains a vibrant democracy and an educated working-age population that can grow our knowledge-based economy in a globalized world.

3.2.1 The Need for Better Information Sharing

Since the passage of the Education Reform Act of 1993, Massachusetts has mandated that all students who are educated with Massachusetts funds participate in standardized Massachusetts Comprehensive Assessment System (MCAS) testing [36]. The tests serve to help improve curriculum and instruction, evaluate student, school, and district performance against state defined standards, and determine student eligibility for the Competency Determination requirement in order to award high school diplomas. Moreover, since 2002, the MCAS tests in Mathematics and English Language Arts (ELA) in grades 3-8 and 10 have satisfied the testing requirements of the No Child Left Behind Act (NCLB) - the latest reauthorization of the Elementary and Secondary Education Act of 1965. The goal of NCLB is to ensure all students in America's public schools are proficient in Mathematics and ELA by the year 2014. Increased accountability means that the distribution of education dollars from Washington and Beacon Hill to schools and districts is contingent upon their students' performance on the tests.

To satisfy the achievement and accountability requirements of the federal legislation, Massachusetts is required to conduct yearly assessments in the aforementioned grades, ensure adequate yearly progress (AYP), and establish that the educators satisfy NCLB's highly qualified teacher (HQT) requirements. A local education authority has to ensure that schools make AYP at the school level and also for specific subgroups of students. The HQT requirement states that teachers must be qualified in the content area they are teaching and that parents be informed if someone who is not qualified is teaching their child [41].

In this three-pronged set up, even if one leaves aside the supplementary federal and state legal requirements, the need for data sharing is obvious. A state agency has to administer the necessary tests and aggregate the results. Then it has to cross-reference this testing data with the student demographic data, which the schools and school districts have, to perform the AYP calculations. Finally, the DESE has to have access to the data bank that contains the educator data - licensure information, educational background, etc. - to meet the HQT requirements. The ability to easily and efficiently integrate these disparate datasets would not only help the state meet NCLB guidelines but might also improve educational outcomes.

3.2.2 Where We Were Then and Where We Are Now

Massachusetts was one of the first states to institute standards based public education reform. The graduating high school class of 2003 was the first one that had to satisfy the MCAS graduation requirement by scoring at the proficient level on the 10th grade Mathematics and ELA tests. In the early days, student demographic data was collected on the MCAS test booklets. As such, DESE was responsible for amassing student non-test data as well the assessment results. The test booklets now contain unique labels for each student so that test administration is made easier on the part of the schools and the districts. The DESE assigned each student a unique ID, called the State Assigned Student ID (SASID), to integrate the demographic and assessment datasets. In the first generation system, the data was a large flat file from which reports were run for accountability measures. Later on, even bigger flat files were created to analyze the performance of students that graduated between 2003 and 2005. Systems were then developed (the data was stored in an Oracle relational system) to automatically merge data and create reports based on a series of business rules. Statisticians at the DESE were responsible for data curation and the accuracy of the data.

In 2003, there were 32 characteristics that made up the student profile, which have been since expanded to 52. Periodically, DESE updates the list of characteristics based on its needs. Currently, the school districts are responsible for collecting the student demographic data and reporting it to DESE. Each district is responsible for securing its own data systems from the vendor of its choice. As there is myriad of systems that exists across the Commonwealth, the school districts are given two to three years to upgrade their systems to have the updated list of characteristics when changes are made.

The Data Warehouse, which is in Phase II and well beyond the flat files, is a Cognos Centric system. It is built as an enterprise education data warehouse.It works well in medium sized school districts, but is not ideal for state-wide data processing and analysis. According to Mr. Robert Bickerton, Associate Commissioner at DESE, various stakeholders - parents, teachers, policymakers, etc. - could ‘make a cup of tea, maybe they go out and have dinner, maybe they could come back the next day, maybe they give up," while they wait for the reports. As a result, the agency is attempting to reengineer the system to make it more efficient. The warehouse aggregates data from school districts three times/year and the educator data is updated once a year. Plans to make the teacher data updates happen as frequently as the student data is in the works. Moreover, proposals have also been made to collect data on student and teacher assignments to classes so that class level, in addition to district and state level, analyses may be performed.

Initially, DESE was only concerned about the scaled scores of students and the levels the scores put them into - advanced, proficient, and needs improvement - for the purposes of accountability reporting. However, in phase II the agency uses an enhanced progress analysis framework called the Growth Model to track student progress. The Growth Model is a student growth percentile model. It is a regression model that creates a cohort of like performing students from all across the state, which is floating and changes from year to year based on test achievement, and not just the school or the district. Starting with a baseline assessment in grade 3, students are compared against their peer group in each subject category. In this system, student performance is not simply the scoring level, but a combination of the level and the student's place in her peer group. The percentile for a school or a subgroup within a school is the mean of the percentile values of the students that make up the group [13].

Even before the growth model, it was easy to see that data sharing was essential for meeting the NCLB reporting requirements. However, the floating cohorts that are made up of students from any school/district in the state and the need to do performance comparisons on a subgroup basis means that without appropriate interoperability and data standards, progress measurements would be difficult, if not impossible.

3.3 A Secure SPARQL Federation for the DESE

The data sharing and interoperability challenges that the DESE faces could be mitigated by using a secure SPARQL federation engine, the architecture of which is described in chapter 4. Such a system, would depend on various SPARQL endpoints that house the information that the federation engine uses.

First, the use of RDF for data representation would make the data collection and storage more efficient. The characteristics that make up a student's demographic profile are updated periodically to satisfy the needs of the agency for reporting and research purposes. Storing this data in tables in relational databases means that all data would have a rigid structure and the database has to be re-architected to make these changes. With RDF, each student's profile could have a unique graph structure. Districts could easily add on new characteristics when the state requests them and choose to either delete the ones that are no longer needed or keep them for the districts' own tracking purposes. Such changes would not require the two or three year time frame that the districts are allotted now, which could lead to faster feedback. Moreover, with such a design, a school can choose to append additional characteristics that are not necessary for state reporting requirements but could provide individualized assistance to students.

The state could specify ontologies that districts have to use to describe the data that is reported to the state. Alternatively, the districts could create ontologies that are most appropriate for their day-to-day use to describe the student and teacher data in their databases. Under such a set up, the district would provide the state with a detailed source description[2]. With appropriate ontology mapping tools, the state and district level systems can be made to interoperate seamlessly.

The availability of a secure federation engine would eliminate the need for the transfer of large amounts of transfer between various databases. The student data can remain on the district servers, and the education licensure information can remain on the DESE Licensure Office's databases. When analyses have to be generated for research or federal reporting purposes, they can be generated by querying the federation engine, which would integrated data from the different databases.

For instance, Mary, an MCAS analyst at the DESE can log in to the federation engine with her credentials to obtain the data necessary to run a performance report for a school. The credentials should allow her access to two data sources to perform a successful query on the federation engine - the student/educator demographic data

[2] Source descriptions are explained in much more detailed in chapter 4

```
PREFI DESE: <http:www.mass.gov/dese#>
SELECT ?student ?mscore ?escore
WHERE
{ ?student DESE:id ?sasid .
           DESE:3rd_math_score ?mscore .
           DESE:3rd_ela_score ?escore .
?sasid     DESE:grade_level \``3''
           DESE:school_name \``Horace Mann''
           DESE:town \``Westtown''
           DESE:race \``Hispanic''
}
```

Listing 3.1: Sample Query

at the district level and the assessment data that is stored on the DESE servers. For example, let us assume she is interested in the performance of third grade students of Hispanic origin in Horace Mann Elementary School in Westtown, Massachusetts. She would issue a query that looks similar to the one in Listing 3.1. She would need to provide appropriate credentials to ensure that she has access to the parts of the DESE database that she is querying.

SPARQL federation is also relevant in the future of education in Massachusetts, and possibly the entire United States. The Massachusetts Executive Office of Education, which is the executive branch agency that is responsible for all levels of education in the state, has plans to observe the progress of students from early childhood, to post-secondary, and eventually employment and wage data to analyze the full impact of the Commonwealth's educational institutions and practices. Such observations are necessary because in the 21st century economy, people constantly have to retool their skill set by engaging in lifelong education. This is evident in the fact that enrollment in adult education programs exceed the rolls of public school students in the Commonwealth [37]. To do this, the federation engine would need to have among its endpoints SPARQL endpoints that contain information from the Department of Early Education and Care, the Department of Higher Education, the Massachusetts Department of Revenue, and others. These agencies would have to develop appropriate ontologies and source descriptions and register with the federator. Each of those endpoints would also specify usage policies that contain access restrictions for users that access the endpoint via the federation engine. The federation engine would be responsible for reasoning over user credentials and policies to ensure that only those users, who have the privileges, access any particular endpoint. Furthermore, the existence of such policies and reasoning would help convince skeptics at these sister agencies that the data that they expose are only made available to those individuals to whom they give access to via inter-agency agreements, which would be made out-of-band.

From a financial perspective, recent developments suggest that there would be adequate resources available to upgrade to a system like this. DESE has received grants, including from the United States Department of Education's Race to the Top funds, to pursue initiatives like the Schools Interoperability Framework (SIF) - a standards framework to which many of the vendors that the districts use are increasingly adhering to [13].

3.3.1 Potential Obstacles

While a secure SPARQL federation engine can boost interoperability among various systems, it is not a silver bullet by any means. There are a number of potential obstacles.

First, Semantic Web technologies are in the early stages of development. Therefore the research literature on them is not as vast as their relational counterparts. Specifically, over thirty years of work has been done on relational database techniques, such as query optimization. Combined with the additional size of data that comes with the expressivity that RDF provides, the fledgling state of SPARQL query optimization means that integration of large education datasets (records for approximately 550,000 students and 80,000 educators [13]) would be painfully slow.

Second, in theory, school districts could use ontologies that best accommodate their needs. However, until data source to domain level mapping techniques, such as the one described in [28] become more robust, DESE might have to prescribe ontologies for all districts to use. This means that a secure SPARQL federator would do no better than the proposed Schools Interoperability Framework [13].

Moreover, a number of non-technological challenges outlined in 3.1.2 are pertinent to DESE. First, as Sharon Wright, the Chief Information Officer in the Executive Office of Education, pointed out, civil servants are reluctant to bring on board better technology because of the fear of finding information that they do not like. If better data organization and management help gleam important insights, it would mean that policymakers would have to go beyond their current responsibilities to rectify the new found problems.

Second, currently there does not exist appropriate legal frameworks to facilitate cross-agency data sharing that would result in longitudinal observations of an individual from early childhood to adulthood and beyond. The educational authority in Massachusetts is dispersed across various agencies, unlike in other states such as Florida, where one executive is legally in charge of all levels and facets of education and training, and has the right to data across the spectrum. Without those frameworks, the Department of Early Education and Care, for instance, could refuse access to the data they possess, under the interpretation of a particular statute. For instance, the major impediment to conducting a longitudinal evaluation is the disagreement over the meaning of section 37 of the Family Educational Rights and Privacy Act[3] [37].

Lastly, as with any large scale data integration effort that the government undertakes, there is fear of Big Brother using the data to turn society into an Orwellian nightmare. Until appropriate safeguards are put in place and the public is satisfied by the effectiveness of those safeguards, elected officials and bureaucrats would face a tough sell in transitioning to a system which has as its integral component a secure SPARQL federation engine.

[3] The full text of this section is in Appendix A.

Chapter 4

A Secure SPARQL Federation Paradigm

Thus far, the various benefits of a secure SPARQL federation systems have been illustrated by comparing it against existing technologies and exploring its application at the Massachusetts Department of Education. Now it is time to look at the SPARQL federation model in more detail. This chapter details the design of the Federation Engine. The features of the various components of the system are described as are the rationale behind various design choices.

4.1 Architecture

As alluded to in chapter 2, the primary motivation of this thesis work was to bring together various Semantic Web technologies and provide an integrated environment that demonstrates the benefits of the secure SPARQL federation paradigm. In addition, this thesis also provides an optimization algorithm for optimizing and rewriting SPARQL queries. The architecture was first laid out in [10]. A few updates have been made to the initial design and the complete architecture is described in the following subsections.

The system is illustrated in Figure 4-1. Its main components are the i) Validator (section 4.1.2), which validates the query provided by the user; ii) the Mapper (section 4.1.3), which splits the query to several subqueries based on descriptions of endpoints; iii) the Optimizer (section 4.1.6), which reorders the subqueries according to the optimization metrics; iv) the Orchestrator (section 4.1.7), which executes the subqueries and integrates the various result sets; and, v) the Proof Generator (section 4.1.8), which generates a proof for each secure SPARQL endpoint based on client supplied credentials and endpoint descriptions, if necessary. The Federation Engine also has in its possession the source descriptions (section 4.1.4) of the endpoints that have registered with it. The Map Generator utility (section 4.1.5) generates a set of mapping rules based on these sources descriptions, which is used by the Mapper. The data in the endpoints are in RDF, which means that query results from multiple endpoints can be easily integrated using common variable bindings.

The system functions as follows: A client submits a query to the Federation Engine on a web-form. The Validator validates the query and forwards it to the Mapper. The Mapper rewrites the query into various subqueries based on the source descriptions known to the Federation Engine. Once the mapping is done, the Optimizer performs the optimization and reorders the subqueries. If any of the endpoints in the query plan requires specific credentials for data access, the user is prompted to supply them at this point. Then, the Proof Generator generates a proof based on the user supplied credentials. The optimized list of subqueries, along with any generated proofs, is forwarded to the Orchestrator. The Orchestrator accepts the optimized list of queries, sends the subqueries along with proofs to the various endpoints, integrates the different result sets, and forwards the final result to the client on the web-form.

Some of the benefits of this architecture have already been presented in chapter 3. The following subsections, which explain the architecture in further detail, elucidate these benefits more.

4.1.1 SSL

The Federation Engine uses the SSL protocol to communicate with a user who wants to query the orchestration engine. The use of SSL ensures that the data transfers between the client and the engine happens via a secure channel. Such transfers would

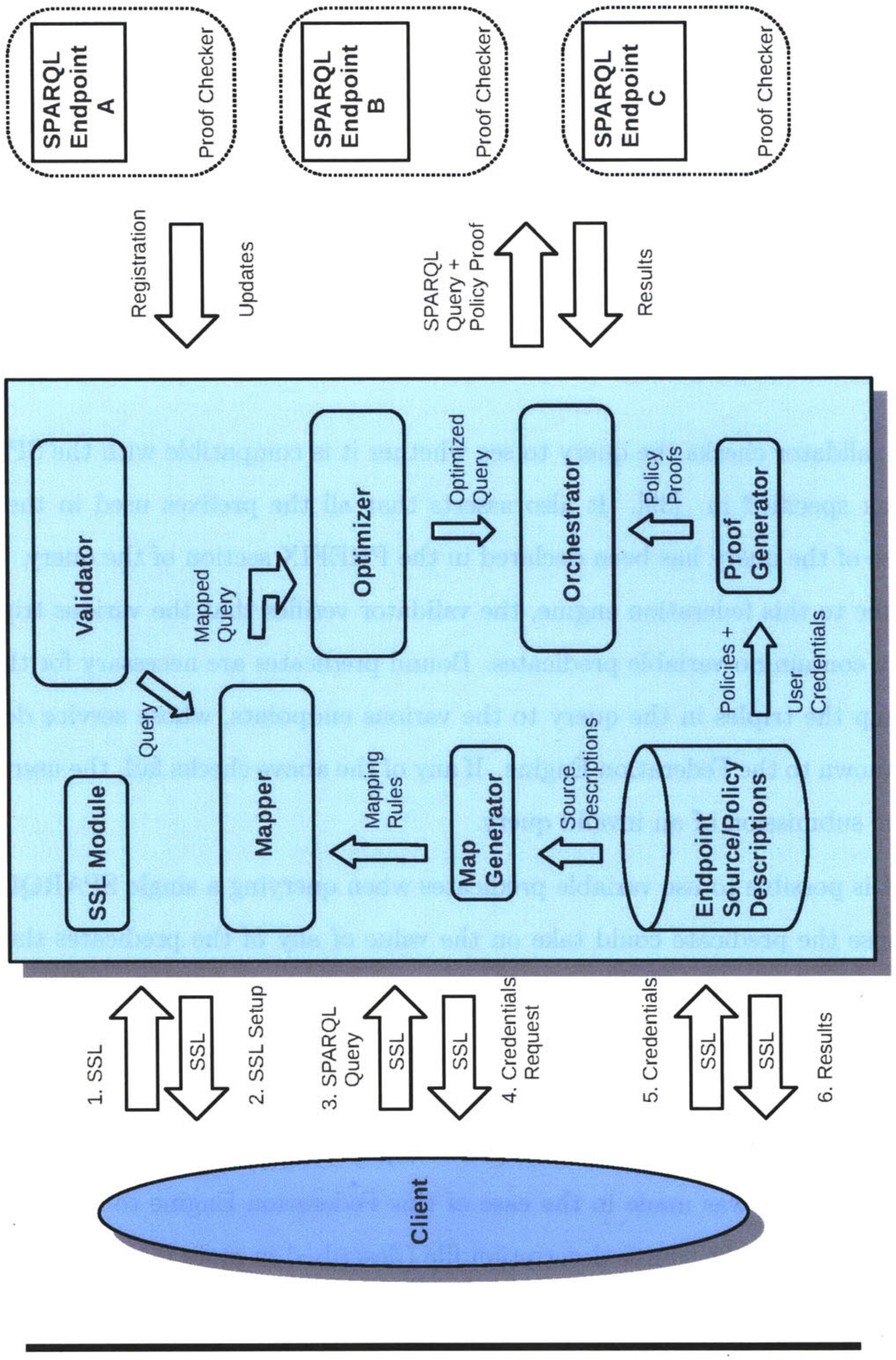


Figure 4-1: System Architecture

protect the confidentiality of queries, results, and credentials that are exchanged. The client contacts the engine through an SSL handshake during which the client provides the engine a certificate with a public key. The SSL module uses the public key to authenticate the user. Once this process is complete, the module notifies the client of its decision. If there was successful authentication, the client is allowed to submit SPARQL queries to the Federation Engine.

4.1.2 Validator

The validator checks the query to see whether it is compatible with the SPARQL 1.0 syntax specified in [39]. It also asserts that all the prefixes used in the WHERE clause of the query has been declared in the PREFIX section of the query. Moreover, specific to this federation engine, the validator verifies that the various triples in the query contain no variable predicates. Bound predicates are necessary for the Mapper to map the triples in the query to the various endpoints, whose service descriptions are known to the Federation Engine. If any of the above checks fail, the user is notified of the submission of an invalid query.

It is possible to use variable predicates when querying a single SPARQL endpoint because the predicate could take on the value of any of the predicates the endpoint contains. However, in a federated setting, variable predicates would mean that the Federation Engine would fail to map a particular query to any endpoint. If subject and/or the object of a triple containing a variable predicate is bound, it may be possible, generally speaking, to map this triple to a particular endpoint. However, a design decision was made in the case of this Federation Engine to have only the list of predicates in the source description file (described in more detail in 4.1.4). While it is possible to find out if an endpoint has a particular predicate based on the bound subject/object information by issuing queries on the fly, such a step would drastically increase the time required to execute a federated query, thereby reducing the utility of this Federation Engine.

```
PREFIX dese: <http://www.mass.gov/dese#>
PREFIX teacher: <http://www.mass.gov/dese/teacher_db#>
SELECT ?student ?mscore ?name ?cert_score ?status
WHERE
{
  ?student dese:id ?sasid ;
           dese:3rd_math_score ?mscore ;
           dese:has_math_teacher ?teacher
  ?teacher teacher:cert_status ?status ;
           teacher:math_score ?score .
  OPTIONAL
  {
    ?teacher teacher:has_name ?name
  }
}
```

Listing 4.1: Input Query

4.1.3 Mapper

The Mapper takes the validated query and generates subqueries based on a predicate to source mapping rules, which are generated by the Map Generator utility. For instance, suppose that the user has input the query in Listing 4.1.

From the set of mapping rules (an example is shown in Listing 4.5), it is evident that the student information is contained in the MCAS endpoint - http://dese.mass.gov/mcas/data/sparql - and the teacher information is found in the educator information endpoint - http://dese.mass.gov/educators/sparql. Based on this information, the Mapper transforms the query into the subqueries in Listing 4.2.

The Mapper puts in the SERVICE annotations, which the Orchestrator uses to select the endpoints to send the subqueries to. Also, notice that the Mapper has resolved the prefixes during the query transformation process.

If the Mapper cannot find one of the query predicates in the set of mapping rules, the user is notified that the query cannot be successful executed. As a result, one could argue that there is some validation done by the Mapper as well.

```
SELECT ?student ?mscore ?name ?cert_score ?status
WHERE
{
  SERVICE <http://dese.mass.gov/mcas/data/sparql>
  {
    ?student <http://www.mass.gov/dese#id> ?sasid .
    ?student <http://www.mass.gov/dese#3rd_math_score> ?mscore .
    ?student <http://www.mass.gov/dese#has_math_teacher> ?teacher .
  }
  SERVICE <http://dese.mass.gov/educators/sparql>
  {
    ?teacher <http://www.mass.gov/dese/teacher_db#cert_status> ?status .
    ?teacher <http://www.mass.gov/dese/teacher_db#math_score> ?score .
  OPTIONAL
  {
    SERVICE <http://dese.mass.gov/educators/sparql>
    {
      ?teacher <http://www.mass.gov/dese/teacher_db#has_name> ?name .
    }
  }
}
```

Listing 4.2: Mapped Query

4.1.4 Source Descriptions

The details of the SPARQL endpoints that are registered with the Federation Engine are stored in the source descriptions. Two sample source descriptions are shown in Listings 4.3 and 4.4. The source descriptions are in the Notation3 (N3)[1] format. Each contains an endpoint's name, URI, read latency, total number of triples, and a list of predicates. For each predicate, the number of triples that contain that predicate as well as the number of distinct objects associated with the predicate are provided. The optimizer makes use of this statistical information. It is possible to generate this statistical information with a set of predefined SPARQL queries. However, this work assumes that they are part of the input from each endpoint that registers with the Federation Engine.

When an endpoint registers with the Federation Engine, it provides a source description. The endpoint administrator also updates the source description when any changes are made to the endpoint's data.

[1] Notation3 is a shorthand non-XML serialization of Resource Description Framework models, designed with human readability in mind [4]

```
@prefix dese: <http://www.mass.gov/dese#>.
@prefix rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>.
@prefix sd: <http://dig.csail.mit.edu/service_description#>.

<http://dese.mass.gov/mcas/data/sparql> rdf:type          sd:Endpoint ;
                                   sd:name
\``MCAS Test Database'' ;
                                   sd:has_policy
<http://dese.mass.gov/mcas/policy.n3> ;
                                   sd:read_latency    5.2;
                                   sd:total_triples   4673239;
                                   sd:num_predicates  3;
                                   sd:has_Predicate   dese:id  ;
                                   sd:has_Predicate   dese:3rd_math_score ;
                                   sd:has_Predicate
dese:has_math_teacher .
dese:id                            sd:num_triples     532346 ;
                                   sd:num_objects     532346 .
dese:3rd_math_score                sd:num_triples     214759 ;
                                   sd:num_objects     100 .
dese:has_math_teacher              sd:num_triples     528631 ;
                                   sd:num_objects     31209 .
```

Listing 4.3: DESE MCAS Endpoint's Source Description

4.1.5 Map Generator

The Map Generator utility generates a set of rules that provides a mapping from the predicates contained in the various source descriptions to the appropriate endpoint. The set of rules generated based on the two source descriptions in Listings 4.3 and 4.4 is illustrated in Listing 4.5. The Mapper uses these rules to rewrite the query with SERVICE annotations for each triple in the query. The set of mapping rules is generated out-of-band and the generation happens when new sources register with the Federation Engine or when an existing source description is updated.

4.1.6 Optimizer

The optimizer takes the mapped query and reorders the SERVICE clauses (subqueries) based on the statistical information contained in the service descriptions.

```
@prefix teacher: <http://www.mass.gov/dese/teacher_db#>.
@prefix rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>.
@prefix sd: <http://dig.csail.mit.edu/service_description#>.

<http://dese.mass.gov/educators/sparql> rdf:type          sd:Endpoint ;
                              sd:name
\``MA Educator Database'' ;
                              sd:policy
<http://dese.mass.gov/educators/policy.n3>
                              sd:read_latency    4.7;
                              sd:total_triples   27931373;
                              sd:num_predicates  3 ;
                              sd:has_Predicate   teacher:has_name ;
                              sd:has_Predicate
teacher:math_cert_score ;
                              sd:has_Predicate   teacher:cert_status .
teacher:has_name              sd:num_triples     77546 ;
                              sd:num_objects     77536 .
teacher:math_cert_score       sd:num_triples     234634 ;
                              sd:num_objects     100 .
teacher:cert_status           sd:num_triples     156450 ;
                              sd_num_objects     5 .
```

Listing 4.4: MA Teacher Information Endpoint's Source Description

The optimization algorithm used by the Optimizer is based on the techniques described in [6]. The purpose of optimization is to minimize the execution time of a federated query. To do this, it is essential to minimize the transfer of result set data from one endpoint to another. The Optimizer ensures that the subquery that would produce the smallest result set is executed first so that the variable bindings from that result set may be used to bind variables in other subqueries, which share the same variables, before they are executed.

Optimization is a two step process. In the first step, the size of the result set of a subquery directed towards a specific SPARQL endpoint is estimated based on the statistical information - number of triples that contain a specific predicate and the number of distinct objects associated with that predicate - contained in the endpoint's source description. Then, the subjects and objects in the triples of a particular subquery are compared against the subjects and objects in triples of all other subqueries to see whether the subqueries share variables. If there are shared variables, it means

```
LABEL 'http://www.mass.gov/dese#id' CONSTRUCT
{?rs <http://www.mass.gov/dese#id> ?ro} {SERVICE
<http://dese.mass.gov/mcas/data/sparql>
{?rs <http://www.mass.gov/dese#id> ro}}
LABEL 'http://www.mass.gov/dese#school' CONSTRUCT
{?rs <http://www.mass.gov/dese#school> ?ro} {SERVICE
<http://dese.mass.gov/mcas/data/sparql>
{?rs <http://www.mass.gov/dese#school> ?ro}}
LABEL 'http://www.mass.gov/dese#reduced_lunch' CONSTRUCT
{?rs <http://www.mass.gov/dese#reduced_lunch> ?ro} {SERVICE
<http://dese.mass.gov/mcas/data/sparql>
{?rs <http://www.mass.gov/dese#reduced_lunch> ?ro}}
LABEL 'http://www.mass.gov/dese#3rd_math_score' CONSTRUCT
{?rs <http://www.mass.gov/dese#3rd_math_score> ?ro} {SERVICE
<http://dese.mass.gov/mcas/data/sparql>
{?rs <http://www.mass.gov/dese#3rd_math_score> ?ro}}
LABEL 'http://www.mass.gov/dese#has_math_teacher' CONSTRUCT
{?rs <http://www.mass.gov/dese#has_math_teacher> ?ro} {SERVICE
<http://dese.mass.gov/mcas/data/sparql>
{?rs <http://www.mass.gov/dese#has_math_teacher> ?ro}}
LABEL 'http://www.mass.gov/dese/teacher_db#has_name' CONSTRUCT
{?rs <http://www.mass.gov/dese/teacher_db#has_name> ?ro} {SERVICE
<http://dese.mass.gov/educators/sparql>
{?rs <http://www.mass.gov/dese/teacher_db#has_name> ?ro}}
LABEL 'http://www.mass.gov/dese/teacher_db#cert_status' CONSTRUCT
{?rs <http://www.mass.gov/dese/teacher_db#cert_status> ?ro} {SERVICE
<http://dese.mass.gov/educators/sparql>
{?rs <http://www.mass.gov/dese/teacher_db#cert_status> ?ro}}
LABEL 'http://www.mass.gov/dese/teacher_db#math_cert_score' CONSTRUCT
{?rs <http://www.mass.gov/dese/teacher_db#math_cert_score> ?ro} {SERVICE
<http://dese.mass.gov/educators/sparql>
{?rs <http://www.mass.gov/dese/teacher_db#math_cert_score> ?ro}}
LABEL 'http://www.mass.gov/dese/teacher_db#cert_date' CONSTRUCT
{?rs <http://www.mass.gov/dese/teacher_db#cert_date> ?ro} {SERVICE
<http://dese.mass.gov/educators/sparql>
{?rs <http://www.mass.gov/dese/teacher_db#cert_date> ?ro}}
LABEL 'http://www.mass.gov/dese/teacher_db#teaches_at' CONSTRUCT
{?rs <http://www.mass.gov/dese/teacher_db#teaches_at> ?ro} {SERVICE
<http://dese.mass.gov/educators/sparql>
{?rs <http://www.mass.gov/dese/teacher_db#teaches_at> ?ro}}
```

Listing 4.5: Mapping Rules for Education

```
Definitions: s: subject; p: predicate; o: object, t: triple;
             a_t: element a of t, bound(a): /variable, for a ∈ $\{s,p,o\}$;
             A.index(a) - value in A that corresponds to 'a'
             num_triples(p) - number of triples containing p;
             num_objects(p) - number of objects with predicate p;
             $E = \bigcup_{i=1}^{n} E_i$, is the set of all endpoints
             $MQ = \bigcup_{i=1}^{n} SQ_{E_i}$ $//SQ_{E_i}$ is the subquery bound for $E_i$
             $S_{E_i} = \bigcup_t s_t, \forall t \in SQ_{E_i}$ is the set of all subjects in $SQ_{E_i}$
             $S = \bigcup_{i=1}^{n} S_{E_1}$
             $O_{E_i} = \bigcup_t o_t, \forall t \in SQ_{E_i}$ is the set of all objects in $SQ_{E_i}$
             $O = \bigcup_{i=1}^{n} O_{E_1}$
             RS_size$_{SQ_{E_n}}$ = $\bigcup_t$ rs_size$_t$, $\forall t \in SQ_{E_i}$
             RS_sizes = $\bigcup_{i=1}^{n}$ RS_size$_{SQ_{E_i}}$

#Ranking Endpoints
Estimate_Result_Set_Size($MQ$)
for each $SQ_{E_i} \in MQ$, do
  for each $atom \in \{S_{E_i}, O_{E_i}\}, \forall i$ do
      for each $\{S_{E_j}, O_{E_j}\}$ ∈ $\{S, O\}$ $\forall j \neq i$ do
          if $atom \in S_{E_j}$ then
             temp = RS_Size$_{SQ_{E_i}}$.index(atom) - RS_Size$_{SQ_{E_j}}$.index(atom)
             if (temp < 0) then
                 RS_Size_Reduce_Ability$_{SQ_{E_i}}$.index(atom) =
                                          max(RS_Size_Reduce_Ability$_{SQ_{E_i}}$)
             else
                 RS_Size_Reduce_Ability$_{SQ_{E_j}}$.index(atom) =
                                          max(RS_Size_Reduce_Ability$_{SQ_{E_j}}$)
sort $\{E\}$ by ($\sum_i$RS_Size_Reduce_Ability$_{SQ_{E_i}}$ \over Read_Latency$_{E_i}$,$\forall i$)
return $\{E\}$

#Estimate_Result_Set_Size($MQ$)
for each $SQ_{E_i} \in MQ$, do
  for each $t \in SQ_{E_i}$ do
    rs_size$_t$ = num_triples(p$_t$)
    if bound($s_t$) then
       rs_size$_t$ = 1 + (1 - #o(p$_t$)/#t(p$_t$)) * num_triples($_t$)
    if bound(o$_t$) then
       rs_size$_t$ = 1
    if $s_t \in S_{E_i}$ then
       rs_size$_t$ = rs_size($S_{E_i}$.$index(s_t)$) = min(rs_size$_t$, rs_size($S_{E_i}$.$index(s_t)$))
    if $s_t \in O_{E_i}$ then
       rs_size$_t$ = rs_size($O_{E_i}$.$index(s_t)$) = min(rs_size$_t$, rs_size($O_{E_i}$.$index(s_t)$))
    if $o_t \in S_{E_i}$ then
       rs_size$_t$ = rs_size($S_{E_i}$.$index(o_t)$) = min(rs_size$_t$, rs_size($S_{E_i}$.$index(o_t)$))
    if $o_t \in O_{E_i}$ then
       rs_size$_t$ = rs_size($O_{E_i}$.$index(o_t)$) = min(rs_size$_t$, rs_size($O_{E_i}$.$index(o_t)$))
  $S_{E_i}$.add($\bigcup_t s_t$), $O_{E_i}$.add($\bigcup_t o_t$), rs_size.add($\bigcup_t$ rs_size$_t$), $\forall t \in SQ_{E_i}$
```

Listing 4.6: Optimization Algorithm

that it is better to execute the subquery that would result in the smaller result set. The optimization algorithm is described in Listing 4.6.

4.1.7 Orchestrator

The Optimizer sends the reordered query to the Orchestrator. The Orchestrator sends each subquery to the endpoint listed after the SERVICE keyword in the subquery. Once the results are returned from an endpoint, the Orchestrator checks to see if any of the subject- or object-bindings in the result set maybe used to bind the variables in the remaining subqueries. If so, these variables are bound before those subqueries are sent to their respective SPARQL endpoints. For those endpoints that require credentials from the querying user, the Orchestrator forwards the policy proofs generated by the Proof Generator using the credentials supplied by the client. Once all the queries have returned with the results, the Orchestrator combines the result sets and returns the combined result set back to the client via the web-form.

4.1.8 Proof Generator

The Proof Generator generates a proof that demonstrates that a given client is privy to the information contained in an endpoint that restricts access to its data to a subset of users. To generate this proof, the Proof Generator makes use of user credentials, provided by the client, and the data usage policies for an endpoint, which the endpoint provides the Federation Engine with (in addition to the source description) during the registration/update process. The endpoint would provide the data usage policies defined in AIR[2], a policy language grounded in Semantic Web Technologies.

The authorization mechanism proposed for the Federation Engine is based on Proof-carrying Authorization (PCA) [2, 3] and earlier work on Policy-Aware Web (PAW) [17, 29, 25, 26] done at DIG. PCA is an authorization framework that is based on a higher-order logic (AF logic) where clients have to generate proofs using

[2] AIR (AMORD In RDF) is a policy language that is represented in Turtle (subset of N3) + quoting and supports AMORD-like constructs. The language describes several classes and properties that can be used to define policies [16]

a subset of AF logic and the authorizing server's task is to check if the proof is grounded and consistent. This allows objects in the system to have a finer-grained control of information and enables a smaller trusted computing base on the server. DIG's work moved these ideas to the open Web using Semantic Web technologies to express policies and proofs.

Though proof generation may be performed by the client, by delegating it to the federation, the load on the client is reduced as are the round trips between the client and secure SPARQL endpoints to obtain required credentials. As a result, if a client does not possess the necessary credentials, she can be notified well before the subqueries are sent off to the endpoints, thereby improving the efficiency of a federated setup.

The Proof Generator is a forward chained reasoner [27] that uses client credentials and online resources to generate a proof for how the client meets a specific policy. This proof is forwarded to the orchestration engine, which uses it to execute subqueries at endpoints that require proofs. If the Proof Generator is not able to generate a required proof based on the client's credentials, the client is informed and is afforded the option to provide additional credentials.

4.1.9 PAW-enabled SPARQL Endpoint

SPARQL endpoints exist as autonomous entities that interact with the secure Federation Engine via SPARQL queries and responses. Each endpoint may, and ideally will, have some data and corresponding descriptions that are unique to it. Any endpoint that wants to operate as part of the architecture described here, would provide the Federation Engine with a source description and, if applicable, usage policies that suit its needs. Those secure endpoints that have usage policies and require proofs from the Federation Engine to allow user access would have a built-in proof checking component [29], which verifies the proofs. If the proof is valid, the endpoint provides the Federation Engine with the results to the query. If not, the Federation Engine is notified of the failed proof check.

If the Proof Generator in the Federation Engine functions properly, it is very unlikely that the proof check would fail. However, the proof check is still necessary because the SPARQL endpoints do not implicitly trust the Federation Engine.

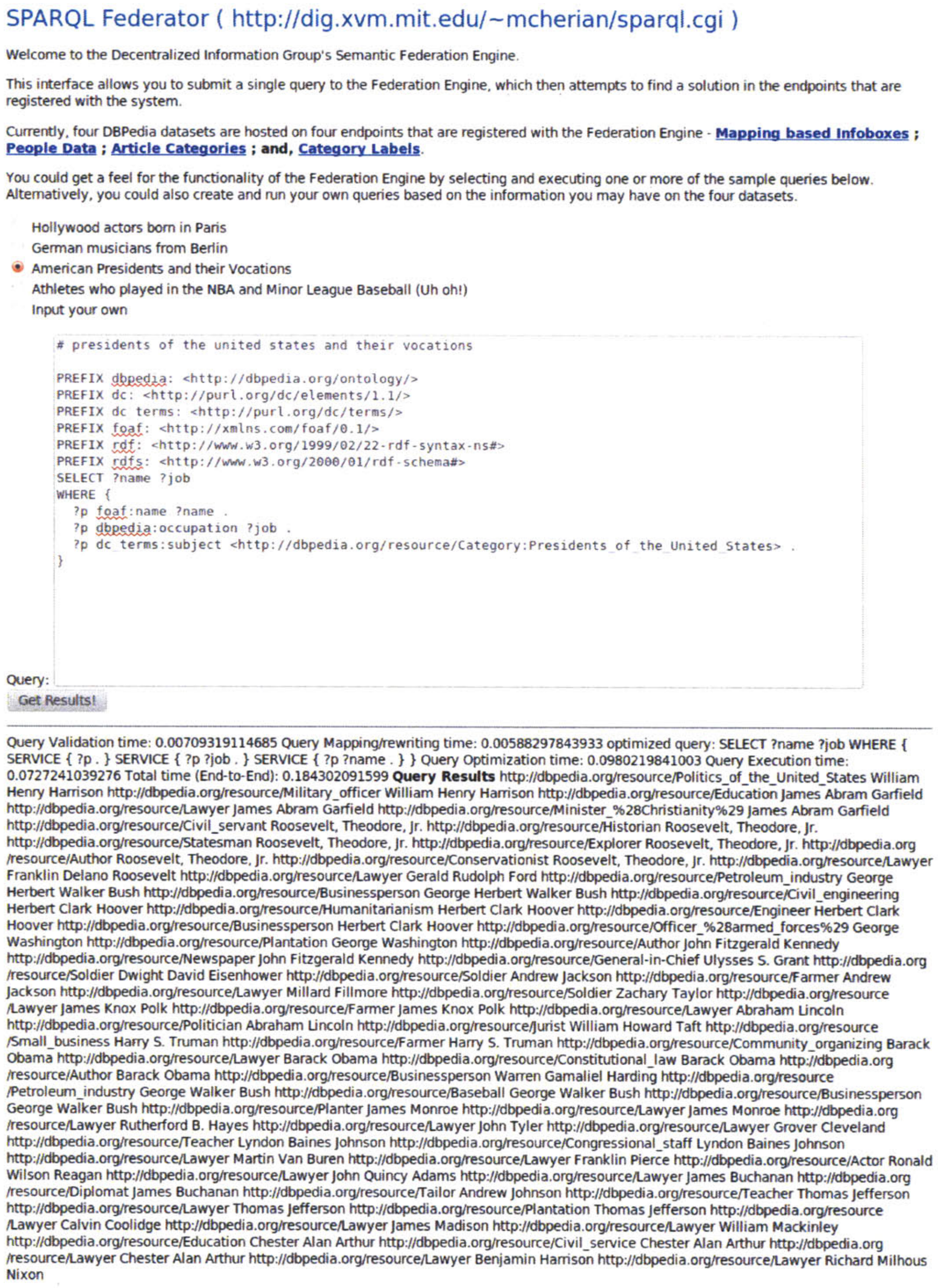
Chapter 5

Implementation

It is now time to describe how the design presented in chapter 4 was put into practice and the roles that various languages and tools played in making the design come to fruition. The functioning system consists of a user interface and the Federation Engine back-end that processes the queries. The Web Interface, described in 5.1 serves as the user interface. The Federation Engine is implemented (described in 5.2) using a combination of C/C++ and Python. The reason for using two languages and a tool that allow them to work in unison is explained in the following subsections. This implementation, however, does not include the proof generation and checking, and the overall policy enforcement mechanisms described in chapter 4. The reasons for this missing functionality is explained in 5.2.1.

5.1 Web Interface

Clients interact with the Federation Engine using a web-form (shown in Figure 5-1) that is written in Python. It has a text box, which allows them to submit a query to the Federation Engine. The result of the query or an explanation for the lack of results is displayed immediately below the query box. The web-form also provides clients with a description of the types of data that are available to the Federation Engine. A handful of sample queries, which give those users who are simply interested in exploring the functionality of the system, are made available as well.



SPARQL Federator (http://dig.xvm.mit.edu/~mcherian/sparql.cgi)

Welcome to the Decentralized Information Group's Semantic Federation Engine.

This interface allows you to submit a single query to the Federation Engine, which then attempts to find a solution in the endpoints that are registered with the system.

Currently, four DBPedia datasets are hosted on four endpoints that are registered with the Federation Engine - **Mapping based Infoboxes** ; **People Data** ; **Article Categories** ; **and, Category Labels**.

You could get a feel for the functionality of the Federation Engine by selecting and executing one or more of the sample queries below. Alternatively, you could also create and run your own queries based on the information you may have on the four datasets.

○ Hollywood actors born in Paris
○ German musicians from Berlin
● American Presidents and their Vocations
○ Athletes who played in the NBA and Minor League Baseball (Uh oh!)
○ Input your own

Query:

```
# presidents of the united states and their vocations

PREFIX dbpedia: <http://dbpedia.org/ontology/>
PREFIX dc: <http://purl.org/dc/elements/1.1/>
PREFIX dc terms: <http://purl.org/dc/terms/>
PREFIX foaf: <http://xmlns.com/foaf/0.1/>
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>
SELECT ?name ?job
WHERE {
  ?p foaf:name ?name .
  ?p dbpedia:occupation ?job .
  ?p dc_terms:subject <http://dbpedia.org/resource/Category:Presidents_of_the_United_States> .
}
```

Query Validation time: 0.00709319114685 Query Mapping/rewriting time: 0.00588297843933 optimized query: SELECT ?name ?job WHERE { SERVICE { ?p . } SERVICE { ?p ?job . } SERVICE { ?p ?name . } } Query Optimization time: 0.0980219841003 Query Execution time: 0.0727241039276 Total time (End-to-End): 0.184302091599 **Query Results** http://dbpedia.org/resource/Politics_of_the_United_States William Henry Harrison http://dbpedia.org/resource/Military_officer William Henry Harrison http://dbpedia.org/resource/Education James Abram Garfield http://dbpedia.org/resource/Lawyer James Abram Garfield http://dbpedia.org/resource/Minister_%28Christianity%29 James Abram Garfield http://dbpedia.org/resource/Civil_servant Roosevelt, Theodore, Jr. http://dbpedia.org/resource/Historian Roosevelt, Theodore, Jr. http://dbpedia.org/resource/Statesman Roosevelt, Theodore, Jr. http://dbpedia.org/resource/Explorer Roosevelt, Theodore, Jr. http://dbpedia.org/resource/Author Roosevelt, Theodore, Jr. http://dbpedia.org/resource/Conservationist Roosevelt, Theodore, Jr. http://dbpedia.org/resource/Lawyer Franklin Delano Roosevelt http://dbpedia.org/resource/Lawyer Gerald Rudolph Ford http://dbpedia.org/resource/Petroleum_industry George Herbert Walker Bush http://dbpedia.org/resource/Businessperson George Herbert Walker Bush http://dbpedia.org/resource/Civil_engineering Herbert Clark Hoover http://dbpedia.org/resource/Humanitarianism Herbert Clark Hoover http://dbpedia.org/resource/Engineer Herbert Clark Hoover http://dbpedia.org/resource/Businessperson Herbert Clark Hoover http://dbpedia.org/resource/Officer_%28armed_forces%29 George Washington http://dbpedia.org/resource/Plantation George Washington http://dbpedia.org/resource/Author John Fitzgerald Kennedy http://dbpedia.org/resource/Newspaper John Fitzgerald Kennedy http://dbpedia.org/resource/General-in-Chief Ulysses S. Grant http://dbpedia.org/resource/Soldier Dwight David Eisenhower http://dbpedia.org/resource/Soldier Andrew Jackson http://dbpedia.org/resource/Farmer Andrew Jackson http://dbpedia.org/resource/Lawyer Millard Fillmore http://dbpedia.org/resource/Soldier Zachary Taylor http://dbpedia.org/resource/Lawyer James Knox Polk http://dbpedia.org/resource/Farmer James Knox Polk http://dbpedia.org/resource/Lawyer Abraham Lincoln http://dbpedia.org/resource/Politician Abraham Lincoln http://dbpedia.org/resource/Jurist William Howard Taft http://dbpedia.org/resource/Small_business Harry S. Truman http://dbpedia.org/resource/Farmer Harry S. Truman http://dbpedia.org/resource/Community_organizing Barack Obama http://dbpedia.org/resource/Lawyer Barack Obama http://dbpedia.org/resource/Constitutional_law Barack Obama http://dbpedia.org/resource/Author Barack Obama http://dbpedia.org/resource/Businessperson Warren Gamaliel Harding http://dbpedia.org/resource/Petroleum_industry George Walker Bush http://dbpedia.org/resource/Baseball George Walker Bush http://dbpedia.org/resource/Businessperson George Walker Bush http://dbpedia.org/resource/Planter James Monroe http://dbpedia.org/resource/Lawyer James Monroe http://dbpedia.org/resource/Lawyer Rutherford B. Hayes http://dbpedia.org/resource/Lawyer John Tyler http://dbpedia.org/resource/Lawyer Grover Cleveland http://dbpedia.org/resource/Teacher Lyndon Baines Johnson http://dbpedia.org/resource/Congressional_staff Lyndon Baines Johnson http://dbpedia.org/resource/Lawyer Martin Van Buren http://dbpedia.org/resource/Lawyer Franklin Pierce http://dbpedia.org/resource/Actor Ronald Wilson Reagan http://dbpedia.org/resource/Lawyer John Quincy Adams http://dbpedia.org/resource/Lawyer James Buchanan http://dbpedia.org/resource/Diplomat James Buchanan http://dbpedia.org/resource/Tailor Andrew Johnson http://dbpedia.org/resource/Teacher Thomas Jefferson http://dbpedia.org/resource/Lawyer Thomas Jefferson http://dbpedia.org/resource/Plantation Thomas Jefferson http://dbpedia.org/resource/Lawyer Calvin Coolidge http://dbpedia.org/resource/Lawyer James Madison http://dbpedia.org/resource/Lawyer William Mackinley http://dbpedia.org/resource/Education Chester Alan Arthur http://dbpedia.org/resource/Civil_service Chester Alan Arthur http://dbpedia.org/resource/Lawyer Chester Alan Arthur http://dbpedia.org/resource/Lawyer Benjamin Harrison http://dbpedia.org/resource/Lawyer Richard Milhous Nixon

Figure 5-1: User Interface

5.2 Federation Engine

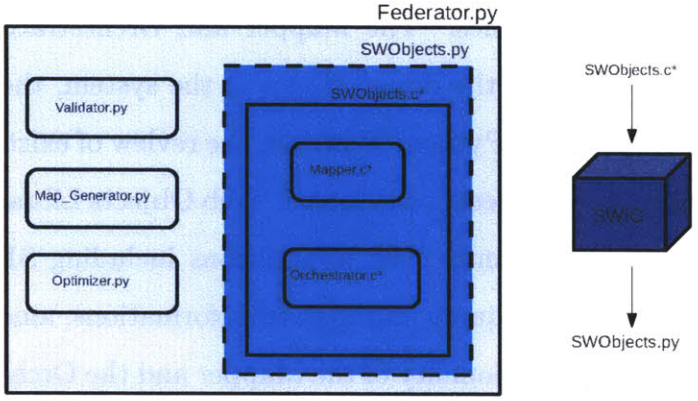
The Validator, Map Generator, and Optimizer are implemented in Python. The source descriptions exist as N3 files. The Mapper and Orchestrator modules are implemented in C/C++. During the design stages of the system, the intention was to implement the entire system in Python. However, the review of existing work in the field led to the discovery of SWObjects - a Semantic Web Objects Library. SWObjects provide general support for Semantic Web applications including SPARQL Query, SPARQL Update, and rule-based query and data transformations, among others [18]. Because SWObjects had the functionality of the Mapper and the Orchestrator, it was efficient to use it rather than reproduce its capabilities from nothing.

The ease of implementing the other modules in Python compared to C/C++ made the language of choice for the rest of the design obvious. However, such a dichotomous setup meant that a way to integrate the components in different languages had to be devised. After comparing a number of tools that integrates Python and C/C++, Simplified Wrapper and Interface Generator (SWIG) was chosen.

5.2.1 SWIG

SWIG is a software tool that allows developers to interface programs written in C/C++ with higher-level languages including Perl, PHP, and Python. With SWIG, programs written in a target language are able to access objects and methods in the C/C++ code as if they were part of a native library [14]. After comparing SWIG with other tools such as Cython and SIP, SWIG was found to have relative advantages against them across two categories. First, SWIG provided better support for native C features than SIP. Second, SWIG allowed the SWObjects code to be used with other high-level language such as Java, Lua, and PHP. While this capability was not necessary for the Federation Engine, the ability to interface SWObjects and Java, for instance, is of great use for the Semantic Web Health Care and Life Sciences (HCLS) Interest Group [1]. As the main developer behind SWObjects, who is a member of

[1] http://www.w3.org/2001/sw/hcls/



Note: .c* donates all necessary C/C++ files (.cpp, .hpp, etc.)

Figure 5-2: Python/SWIG/C

HCLS, was a collaborator on this thesis work, the use of SWIG would turn out to be mutually beneficial. Hence, SWIG was chosen as the tool for integrating the existing C/C++ code with the new code that would be implemented in Python.

Figure 5-2 illustrates how SWIG facilitates the use of SWObjects - implemented in C/C++ - for the rest of the Python code. SWIG generates a Python wrapper that encapsulates the SWObjects code. The SWIG processing is quite time consuming and computationally taxing. However, this takes place offline and once SWIG-processing is done, SWObjects.py exists as a standalone Python module than can be imported into any other Python module. No further SWIG-processing is required at run time.

Although without SWIG a lot of code might have had to be rewritten in Python, in hindsight the decision to use SWIG might not have been the ideal one. While, it has the capacity to make most features of C and C++ available in Python, there was often little information available for using it for purposes beyond the very simple use cases. This lack of detailed documentation meant that it took much longer than expected to make SWIG operational for this project. As a result, there was no time left for the implementation of the policy verification and proof generation mechanisms. Hence, in the current implementation, the Federation Engine can only query open SPARQL endpoints that do not have any access restrictions. However, implementing the policy reasoning framework should not be difficult as the system contains hooks for calling proof generation and checking components.

5.2.2 Fyzz and RdfLib

The Python implementation would not have been possible without two libraries, which have been developed for Semantic Web Technologies. Fyzz[2] is a Python parser for SPARQL. Fyzz was used for implementing the Validator. Because the latest version (0.1.0) of Fyzz is compatible with the SPARQL 1.0, it was necessary to update the Fyzz SPARQL grammar to include the SPARQL 1.1 features that the Federation Engine uses. Specifically, this meant updating the grammar so that Fyzz can handle SERVICE keywords.

RdfLib[3] is a Python library for working with RDF. It includes parsers and serializers for RDF/XML, N3, NTriples, Turtle, Trix and RDFa. For the purposes of implementing the Federation Engine, the N3 parser in RdfLib was used to extract information from the source description files.

[2] http://www.logilab.org/project/fyzz

[3] http://www.rdflib.net/

Chapter 6

Evaluation

This chapter details the tests that were designed to evaluate the Federation Engine as well as discusses the Engine's performance on these tests. Section 6.1 details the rationale behind the choice of test queries, their coverage, and the endpoints whose data were used for running the test queries. The Discussion section (6.4) provides some explanations for the test results.

6.1 Test Plan

As presented in chapter 4, the novelty of this system stems from the integration of various Semantic Web technologies as well as the Optimizer. The evaluation assesses the system's performance on these two distinct aspects. The metric used to evaluate the system is the time it takes to perform various task associated with the system. For the purposes of assessment, five distinct time metrics were identified: Validation time - the duration from when the client submits the query to when the Validator sends the query to the Mapper; Mapping time - the time it takes for the Mapper to perform its functions; Optimization time - the time it takes for optimization, Transformation time - the sum of Mapping time and Optimization time; Execution time - the time it takes for the Orchestrator to take an optimized query, execute it against the various endpoints, and recombine the various result sets; and Total time - the end-to-end time. All times were calculated using the time function in Python's time module.

For all tests, queries that would generate three subqueries based on the mapping rules were used. Two different types of tests were conceived; the first to evaluate the Optimizer, and the second to evaluate the overall system. Every test was run three times and the average time was calculated. The Federation Engine was hosted on a 64-bit virtual machine running Ubuntu 10.04 Lucid Lynx Server with 512 Megabits (MB) of Random Access Memory (RAM) and 10 Gigabytes (GB) of hard disk space.

6.2 Optimizer Tests

The metrics used to assess the optimizer were Mapping Time, Optimization Time, and Transformation time. The Optimizer functions independently of query execution and only makes use of the endpoint source descriptions to do the optimization. As such, all optimizer tests were performed using a set of mock source descriptions, which were auto-generated using a python script. A sample mock description is presented in Appendix B.

```
SELECT ?s1 ?s2 ?s3
WHERE {
?s1 p1 ?o1
?s2 p2 ?o2
?o2 p3 ?o3
}

Note: p1, p2, and p3 were replaced in each test with the
      predicates in the auto-generated source descriptions.
```

Listing 6.1: Optimization Test Query

6.2.1 Number of Predicates vs. the Number of Triples per Predicate

The first set of tests involved varying the number of predicates and the triples. Three endpoints, each with a total of 300,000 triples were used for this test. A test was run

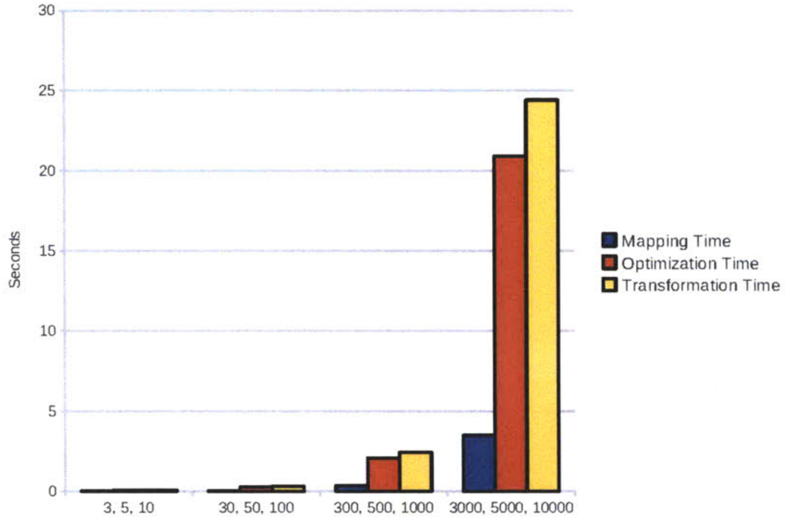


Figure 6-1: Number of Predicates vs. Number of Triples per Predicate

such that the sum of the products of the number of predicates in an endpoint and the number of triples associated with each predicate added up to 300,000. For instance, if an endpoint had 30 predicates, it would have 10,000 triples associated with each endpoint. Four tests with the number of predicates at each endpoint ranging from 3, 5, and 10 to 3000, 5000, and 10000 (across the three endpoints, and the corresponding number of triples per endpoint such that the total number of triples would be 300,000) were run. For each test iteration, the query in Listing 6.1 was mapped and optimized.

The results for this test are shown in Figure 6-1. Increasing the total number of triples in an endpoint and the number of triples per predicate for each of the four cases tested above did not have any noticeable impacts on the Mapping and Optimization times.

```
#Variables - No Sharing
SELECT ?s1 ?s2 ?s3
WHERE {
?s1 p1 ?o1
?s2 p2 ?o2
?s3 p3 ?o3
}

#Variables - Some Sharing
SELECT ?s1 ?s2 ?s3
WHERE {
?s1 p1 ?o1
?o1 p2 ?o2
?s3 p3 ?o3
}

#Variables - Full Sharing
SELECT ?s1 ?s2 ?s3
WHERE {
?s1 p1 ?o1
?o1 p2 ?o2
?o2 p3 ?s1
}
```

Listing 6.2: Variable Sharing between Subqueries

6.2.2 Sharing of Variables between Subqueries to Different Endpoints

Next, the impact of different types of queries on the optimizer was assessed. For this, the three source descriptions with 300, 500, and 1000 predicates, respectively, were used. The following cases were tested: Variable subjects and objects with no variables sharing between endpoints; Variable subjects and objects with two endpoints sharing variable; and, Variable subjects and objects with variable sharing between all endpoints. The three test cases are illustrated in Listing 6.2.

The results are shown in Figure 6-2. Similar tests were conducted with bound subjects and objects. The queries are in Listing 6.3. Notice in the queries that the values for subjects and objects need not be present in the endpoint for successful optimization. The optimizer only cares whether the subjects and objects are variables or not. It has no knowledge of the contents of an endpoint beyond the predicates in

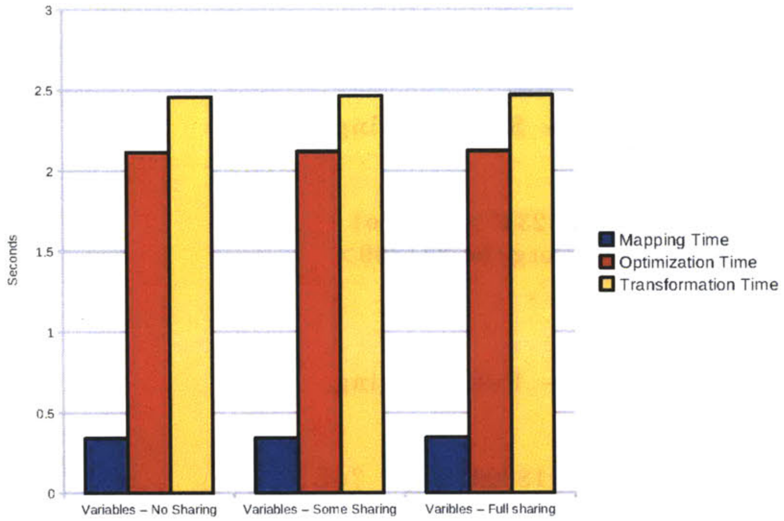


Figure 6-2: Variables Sharing between Service-Subqueries

the source descriptions. A separate graph for the results are not provided because the results were identical to the tests with variable subjects and objects.

6.2.3 Number of Triples

Three tests were run to observe whether increasing the total number of triples and the number of triples per predicate, while holding the number of predicates constant, had any effect on optimization times. This time the three source descriptions with 30, 50, and 100 predicates, respectively, were used.

First, tests were run with 30,000, 300,000, 3,000,000, 30,000,000, and 300,000,000 total triples, and the corresponding number of triples per predicate. For example, the 50 predicate endpoint had 600,000 triples per predicate for the test with total triples of 30,000,000. The results are shown in Figure 6-3.

```
#Bound Subjects/Objects - No Sharing
SELECT ?s1 ?s2 ?s3
WHERE {
<http://www.alpha.org/as:193856> p1 ?o1
?s2 p2 <http://www.alpha.org/as:193856>
?s3 p3 ?o3
}

#Bound Subjects/Objects - Some Sharing
SELECT ?s1 ?s2 ?s3
WHERE {
<http://www.alpha.org/as:234?> p1 ?o1
?o1 p2 <http://www.beta.org/bo:341209>
?s3 p3 ?o3
}

#Bound Subjects/Objects - Full Sharing
SELECT ?s1 ?s2 ?s3
WHERE {
<http://www.alpha.org/as:183095> p1 ?o1
<http://www.beta.org/bs:IGe23> p2 <http://www.alpha.org/as:183095>
?o2 p3 <http://www.beta.org/bs:IGe23>
}
```

Listing 6.3: Variable Sharing between Subqueries (Bound Subjects/Objects)

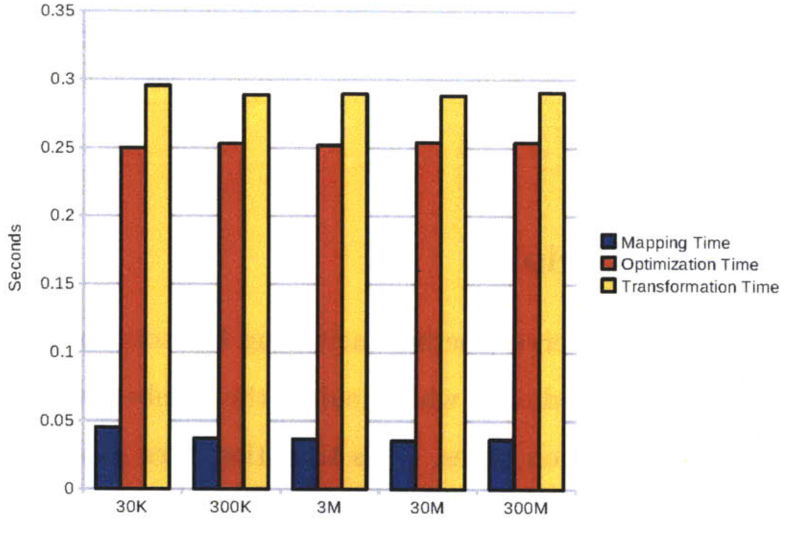


Figure 6-3: Total Number of Triples/Number of Triples per Predicate

Table 6.1: Datasets for Federation

Endpoint	Dataset (Number of Triples)
E1	DBPedia Person Data (1.7 million)
E2	DBPedia Article Categories (12 million)
E3	DBPedia Category Labels (632,000)
E4	DBPedia Infoboxes (13.8 million)

6.3 Federation Tests

A set of tests were run to assess the end-to-end performance of the Federation Engine and to test whether the optimization had worked. All time metrics described in section 6.1 were measured in these tests. As in the case with the Optimizer tests, queries that would generate three or more subqueries were used to run these tests. However, unlike the Optimizer tests, there needed to be actual endpoints and datasets to execute the subqueries against. The motivation for the type of queries and the datasets were taken from [40].

Four SPARQL endpoints [1] were set up using 4store [2] - a scalable RDF database - for testing the performance of the Federation Engine. They were populated with datasets from DBPedia [3]. The 4store endpoints are used are shown in Table 6.1. E1, E2, and E3 are hosted on 64-bit virtual machines(VM) running Ubuntu Server 10.04. E1 is hosted on a VM, which has 4 GB of hard drive space and 256 MB of RAM. E2 and E3 are co-hosted on a single VM, which has 10 GB of hard drive space and 256 MB of RAM. E4 is hosted on a server (details...). The source descriptions for these endpoints were created by issuing a number of queries, the results to which contained predicate and statistical information, directly at the endpoints.

To compare unoptimized query execution against optimized query execution, a duplicate of the Web Interface was created. The source code of the two interfaces were identical, except for the the set of subqueries that was executed (optimized for optimized execution and vice versa).

[1] http://mcherian.xvm.mit.edu:8000/sparql, http://matminton.xvm.mit.edu:9000/sparql, http://matminton.xvm.mit.edu:9090/test/, and http://air.csail.mit.edu:9000/sparql

[2] http://4store.org/

[3] http://wiki.dbpedia.org/Downloads36

Four queries (shown in Listings 6.4, 6.6, 6.7, and 6.5) of varying difficulty were created. These were submitted to two instances of the Web Interface.

```
# German Musicians who were born in Berlin
PREFIX dbpedia: <http://dbpedia.org/ontology/>
PREFIX dbp_resource: <http://dbpedia.org/resource/>
PREFIX dbp_category: <http://dbpedia.org/resource/Category:>
PREFIX dc_terms: <http://purl.org/dc/terms/>
PREFIX foaf: <http://xmlns.com/foaf/0.1/>
PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>
SELECT ?person ?name ?birthday
WHERE
{
  ?person foaf:name ?name .
  ?person dbpedia:birthDate ?birthday .
  ?person dbpedia:birthPlace dbp_resource:Berlin
  ?person dc_terms:subject dbp_category:German_musicians .
  OPTIONAL {
  dbp_category:German_musicians rdfs:label ?label .
  }
}
```

Listing 6.4: Q1: German Musicians Who Were Born In Berlin

```
# Presidents of the United States and their Vocations
PREFIX dbpedia: <http://dbpedia.org/ontology/>
PREFIX dbp_category: <http://dbpedia.org/resource/Category:>
PREFIX dc_terms: <http://purl.org/dc/terms/>
PREFIX foaf: <http://xmlns.com/foaf/0.1/>
SELECT ?name ?job
WHERE {
  ?person foaf:name ?name .
  ?p dbpedia:occupation ?job .
  ?p dc_terms:subject dbp_category:Presidents_of_the_United_States .
}
```

Listing 6.5: Q4: Presidents of the United States & Their Vocations

```
# People who played professional baseball and basketball
PREFIX dbpedia: <http://dbpedia.org/ontology/>
PREFIX dbpedia_category: <http://dbpedia.org/resource/Category:>
PREFIX dc: <http://purl.org/dc/elements/1.1/>
PREFIX dc_terms: <http://purl.org/dc/terms/>
PREFIX foaf: <http://xmlns.com/foaf/0.1/>
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>
SELECT ?name
WHERE
{
  ?p dc_terms:subject dbpedia_category:Minor_league_baseball_players .
  ?p foaf:name ?name .
  ?p dc_terms:subject dbpedia_category:American_basketball_players .
}
```

Listing 6.6: Q4: Athletes Who Played Professional Baseball & Basketball

```
#Paris born Movie Stars
PREFIX dbpedia: <http://dbpedia.org/ontology/>
PREFIX dbp_resource: <http://dbpedia.org/resource/>
PREFIX dc: <http://purl.org/dc/elements/1.1/>
PREFIX foaf: <http://xmlns.com/foaf/0.1/>
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>
SELECT ?name ?m
WHERE
{
  ?m dbpedia:starring ?p .
  ?p dbpedia:birthPlace dbp_resource:Paris .
  ?p foaf:name ?name .
  FILTER (?city = dbp_resource:Paris)
}
```

Listing 6.7: Q3: Hollywood Actors Who Were Born In Paris

These queries were selected for the varying sizes of the result sets they produced and the number and combinations of endpoints the subqueries had to be sent off to during execution.

It was noted that the Validation and Mapping Times were identical while running optimized and unoptimized versions of all four queries. This was in line with expectations as the Federation Engine code as well as query structures involved in both cases are the same.

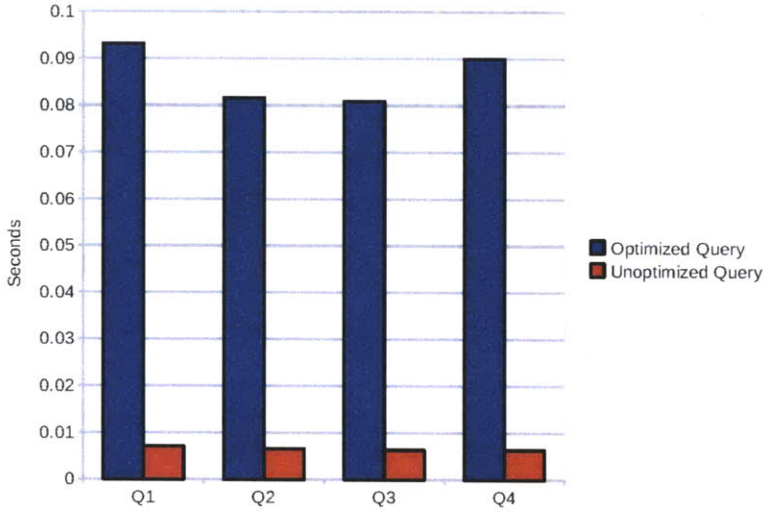


Figure 6-4: Transformation Times

Table 6.2: Queries, Endpoints, & Result Set Sizes

Query	Endpoints	RS Size(expected)	RS Size(optimized)	RS Size(unoptimized)
Q1	E1, E2, E3	1	1	1
Q2	E1, E4	1	1	1
Q3	E2, E4	8	8	8
Q4	E1, E2, E4	69	69	1

Table 6.2 shows all the endpoints needed to answer a query, the expected size of the result set, and the sizes of the result sets obtained from the execution of the optimized and unoptimized versions of the query. These numbers were calculated by manually issuing the subqueries to the endpoints and tabulating the size of the result set.

The expected result sizes and the actual result sizes are included to track a behavior often seen in most well-established SPARQL endpoints. Such endpoints limit the number of results it outputs to prevent poorly designed queries from consuming a large amount of system resources. Therefore, even when an unoptimized query is

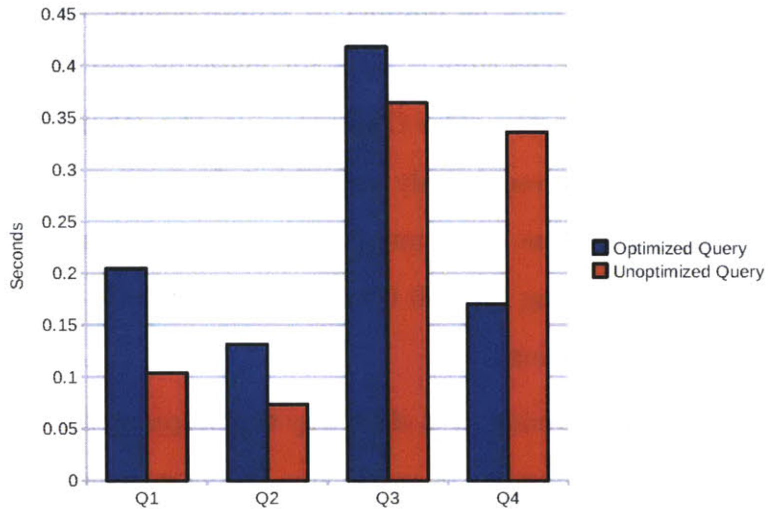


Figure 6-5: Total (End-to-End) Times

fired against an endpoint, a result set is returned in a reasonable amount of time, instead of a hung query. However, because of the limit on the results, the returned result set would only be a subsection of the amount of data that matches the Basic Graph Pattern in the query. During testing, this limit could have been done away with as the endpoints were set up purely for testing purposes. However, the decision was made to leave the limits in place to simulate real-world behavior.

Figure 6-4 compares the Transformation times of the optimized and non-optimized versions. Figure 6-5 compares the Total time between the two.

6.4 Discussion

6.4.1 Optimizer Tests

The results of the Optimizer tests provide several insights. As seen in Figure 6-3, the total number of triples and the number of triples per predicate had no impact on the Mapping or Optimization times. This result was indeed in line with pre-test expectations. The mapping rules contain no statistical information; hence varying the statistical information such as number of triples should not have any effect on the Mapping time, which is the time it takes for mapping and rewriting the query by scanning the mapping rules file. The Optimizer, on the other hand, does make use of endpoint statistics. However, it only uses this information to do basic arithmetic operations to rank the various endpoints. Therefore, it was expected that the range of the numbers used in testing (30,000 to 300,000,000) should not have any effect on the time taken for optimization.

By the above line of reasoning, it is clear why the Mapping times and Optimization times increased when the number of predicates increased (see Figure 6-1). As the number of predicates increases, both the Mapper and the Optimizer have to read in and parse larger mapping rules file and source descriptions, respectively, to perform their functions.

Furthermore, the number of bound variables in queries and the level of sharing of variables between subqueries had little impact on the timing values, as seen in Figure 6-2. First of all, the Mapper functions are independent of these two characteristics. And, once again, it was not surprising that the optimization times were unaffected. Regardless of whether the variables are bound or not, and irrespective of whether two or more subqueries share variables, the optimizer performs the same number of checks and iterations when it parses a query with SERVICE bound subqueries. Hence, the extent of sharing of subjects and objects, and the number of bound subjects and objects would not have any impact on the optimization times.

The various Optimizer tests reveal that increasing the size of the mapping and source description files have a significant impact on mapping and optimization times. Varying other aspects of queries and endpoints had little effects on said times. Therefore, it is possible that decreasing the scan/parse times for the file would have significantly improve the performance of the Federation Engine on the whole. The use of a faster file I/O library could go a long way towards achieving this goal. However, this possibility was not explored in this research.

6.4.2 Federation Tests

The results of the Federation tests were more interesting than the ones from testing the Optimizer by itself. The obvious observation was the fact that Transformation times (Figure 6-4) were noticeably higher across all four queries for the Engine with the Optimizer when compared to the one without it. There is no surprise here because the Transformation time is the sum of Mapping time and Optimization time, and Optimization time is zero for the federator without the optimizer. However, notice that the Transformation time is only a fraction of the Total time.

A first glance at Total times (Figure 6-5) gives the impression that the unoptimized federator did better than the optimized one in 3 out of 4 queries. However, a closer look reveals some interesting outcomes. As the complexity of the query increased, the Total time advantage that the unoptimized version has markedly disappear. While the unoptimized version did 50% better than its optimized counterpart in Q1, by Q3 the advantage diminishes to about 13%, and in Q4 the optimized query takes 50% to return the result set after query submission.

The other key observation from the Federation tests is seen by comparing the Total time (Figure 6-5) to the expected and actual number of results (Table 6.2). Note that for Q4, the expected size of the result set was 69. While the optimized version, did return all 69, the unoptimized version returned a result set of size 1. What happened here was that the federator without the optimizer sent the subquery that contains the triple *?p dbpedia:occupation ?job* in its WHERE clause to be executed first. However, because this subquery has a low selectivity (with a result set size of approximately 140,000), the endpoint hit its upper bound and returned only a subset of the results. And, this result set only had one U.S. President in it, which caused the conjunctions with the subsequent subqueries to return exactly one result.

The Federation tests demonstrate that, although the addition of the Optimizer contributes to an increase in Total time, this increase is a price worth paying. Even with a marginal increase in the complexity, the disadvantage of having the Optimizer starts to disappear and we also see that without it in line, incomplete result sets would be returned.

Another lesson learned from testing relates to SPARQL endpoints. Initially, the plan was to use existing SPARQL endpoints to run the queries. Three or four endpoints, which shared Graph elements, would be chosen and queries designed such that the Mapper would generate subqueries that hit three or more of them. Choosing popular existing endpoints would also buttress the practical relevance of the Federation Engine. Moreover, it would have involved not repeating work that is not fundamental to the design or implementation of the Federation Engine. As a result, a few of the Bio2RDF endpoints [4] were identified. However, the task of identifying the overlap between Graphs on different endpoints was more daunting than initially expected. The large size of the datasets meant that the queries required to extract endpoints' predicate and statistical information required to create the source description files did not run to completion.

Both the Optimizer and Federation tests are very pertinent to the proposed application of the Federation Engine at the Massachusetts Department of Education in chapter 3. The database that holds student MCAS information, for instance, would contain approximately 12 millions triples [5] of raw test data. School districts would maintain student demographic information databases that contain, on average, approximately 75,000 triples [6]. The test demonstrate that even if the number of triples were to increase an order of magnitude, the Optimizer would continue to perform sufficiently. Moreover, the hardware that hosts the education datasets would be much more powerful and feature-rich than the virtual machines used for evaluating the Federation Engine.

However, the effects of the size of data transferred between endpoints on federation capabilities are still unknown. A query similar to the one described in Listing 3.1 might involve tens or even hundreds of school endpoints. Even in those cases where the number of endpoints is in the single digits, the data transferred might be much larger

[4] http://www.freebase.com/view/user/bio2rdf/public/sparql

[5] 550,000 students * 3 tests/year * 7 testing years = 11.55 million

[6] (550,000 students/391 districts) * 52 student characteristics = 73,146

than what was encountered during testing. For instance, MCAS math scores from all previous years of all tenth grade students from a particular school (approximately 1800 [7]) might have to be pulled to observe an interesting pattern. The Federation Engine's performance under such circumstances cannot be assessed until a commercial scale test system is deployed at the DESE and in districts around the Commonwealth.

6.5 Limitations

Although the Optimizer and the Federation Engine as a whole proved quite successful in the various tests, this system has a few shortcoming. First, if a user wished to execute queries beyond the sample ones provided on the Web Interface, she would find it very difficult to do so. It would require knowledge of the underlying ontology and the predicates contained in the endpoints registered with the Federation Engine. However, currently this information is not being furnished to the user in any form.

Second, the Optimizer is based on an algorithm that makes use of a simple cost model. This set up would prove insufficient for queries that are much more complex than the ones explored in testing.

Third, the requirement for bound predicates limits the types of queries that a user is able to execute using the Federation Engine. This limitation can prevent the engine from gaining a large user base.

[7] 300 students * 6 years = 1800 data points

Chapter 7

Summary

The ability to query distributed heterogeneous data sources and aggregate information across them is a key requirement for realizing the full potential of the Semantic Web. In this thesis, I presented the design, implementation, and a relevant application of a secure SPARQL federation engine that takes a big step in that direction. First, I detailed the current research in the field and the existing technologies that made this engine possible. Then, I discussed a specific use case for this system, which illustrated the importance of this system for a large public sector agency. After this, I described the architecture of the system, paying particular attention to design choices. The system was implemented using a variety of software tools, including a C/C++ wrapper for Python, which although very useful, was a major cause for the lack of security mechanisms in the final version of the system. The Federation Engine underwent a series of tests - both at the system level and at the level of novel components, which showcased the engine's capabilities and a few limitations. Section 7.1 highlights the key contributions of this thesis work and section 7.2 identifies a few areas of research that can improve the performance and features of this novel system.

7.1 Contributions

- **Designed and implemented an end-to-end SPARQL Federation Engine**

Although the Semantic Web had made many advances over the past decade, there did not exist an integrated secure environment to perform federated queries to mash up data from distributed heterogeneous data sources. I designed the architecture for a secure Federation Engine that can aggregate information from multiple SPARQL endpoints, while adhering to appropriate usage policies. I also implemented the said architecture, albeit without the security layer, by using existing Semantic Web tools such as the Fyzz and RdfLib Python libraries, and SWObjects - a C/C++ Semantic Web objects library.

- **Developed query optimizer that facilitates efficient query execution**
  I also developed an Optimizer that reorders subqueries, which are directed at different endpoints, to ensure efficient and complete query execution. The Optimizer depends on the endpoint statistics based cost model described in [6]. Without the Optimizer, all queries, but the ones with very low complexity, would fail to execute in practical settings. This is because most SPARQL endpoints that exist today have a low upper bound on the number of results they would return to any query to prevent the exploitation of system resources. Given this status quo, subqueries, which have low selectivity, if executed first (as would be the case without the Optimizer) would return incomplete or NULL result sets.

- **Initiated the Use and Integration of SWIG into the SWObjects project**
  SWObjects is a large open source C/C++ code base that provides general support for Semantic Web applications including SPARQL queries, SPARQL updates, and rule-based query and data transformations, among others. Research Groups, notably the Semantic Web Health Care and Life Sciences Group (HCLS), is interested in using SWObjects library for many of their Web 3.0 applications. However, HCLS uses Java for their research activities. In fact, many research undertakings these days use higher level languages, such as Java and Python. My thesis work spearheaded the inclusion of SWIG - a C/C++ wrapper for higher level languages - in the SWObjects project so that the Federation

Engine could make use of some of the features of SWObjects. The work done on interfacing SWIG and SWObjects is invaluable for HCLS' and other projects that depend on the SWObjects library.

In addition to the Python/C++ back-end that facilitates query federation, this research has also produced a web-form that users can use to run a custom query or a range of queries, that exhibit the Federation Engine's capabilities. Also integrated into the end-to-end system is a framework that makes available to a user the various time metrics associated with the execution of a federated query.

7.2 Future Work

Although the prototype of the Federation Engine breaks new ground on many fronts, a number of features and improvements can better the current state of the system. The first obvious feature addition is the policy reasoner and checker that were described in chapter 4. The Proof Generator and Checker components, as described in [27] and [29], respectively, and implemented as part of the Policy Aware Web project completed by DIG, may easily be integrated into the Federation Engine. The fact that both systems are implemented in Python should make this integration seamless.

Second, advances need to be made to automate the generation of source descriptions. For the purposes of the Federation Engine, source description files had to be manually created using the results of queries, which were formulated to obtain predicate and statistical information, issued to each individual endpoint. This process was cumbersome, and, moreover, impractical for endpoints that have a large number of predicates. A tool, which can automatically generate the source description files using the results of a number of predetermined queries, would facilitate easier registration of new endpoints and updates of existing ones. The end result of developing and issuing such a tool would be an increase in the range of queries the Federation Engine can provide results for.

Third, for demonstrating the capabilities of the Federation Engine, four SPARQL endpoints were set up with subsets of DBPedia data. However, the practical relevance of the engine would become much more evident once more persistent SPARQL endpoints, whose Graphs overlap, are present on the Web. These endpoints must also provide easy interfaces for a tool such as the automatic source description generator to obtain relevant predicate and statistical information.

Fourth, in the current implementation, mapping of subqueries to endpoints are done before optimization. The performance of the Federation Engine may improve dramatically if the Mapper is implemented as a sub-component of the Optimizer. For example, in such an arrangement, mapping to a particular endpoint maybe avoided altogether if the optimizer determines that the cost of querying that endpoint with a particular subquery is prohibitively high.

Fifth, much work has been done on optimization techniques for relational database federations. Until either those techniques are made applicable to Semantic Web data models, or current research into SPARQL optimization progresses further, the Federation Engine would not be able to handle very complex queries, the successful execution of which are essential for the SPARQL federation paradigm to become a key player among data integration technologies.

Lastly, as with most applications on the web, the user base of the Federation Engine would grow dramatically with advancements in natural language processing. The ability of clients to issue natural language queries and obtain results from a large number of endpoints could considerably increase the mass appeal of the Semantic Web.

Appendix A

Family Educational Rights and Privacy Act, Section 37

20 U.S.C. 1232g; 34 CFR Part 99, Section 37

What conditions apply to disclosing directory information?

(a) An educational agency or institution may disclose directory information if it has given public notice to parents of students in attendance and eligible students in attendance at the agency or institution of:

(1) The types of personally identifiable information that the agency or institution has designated as directory information;

(2) A parent's or eligible student's right to refuse to let the agency or institution designate any or all of those types of information about the student as directory information; and

(3) The period of time within which a parent or eligible student has to notify the agency or institution in writing that he or she does not want any or all of those types of information about the student designated as directory information.

(b) An educational agency or institution may disclose directory information about former students without meeting the conditions in paragraph (a) of this section.

Appendix B

Test Source Description

```
@prefix sd: <http://dig.csail.mit.edu/service_description#>.
@prefix rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>.
@prefix alpha: <http://www.alpha.org/ns#>.

<http://www.300k_3_100k.com/sparql>   rdf:type     sd:Endpoint ;
                                      sd:name      ”Mock Description” ;
                        sd:description  ”Created for Running Tests” ;
                        sd:read_latency 4 ;
                        sd:total_triples  300000 ;
                                      sd:num_predicates 3 ;
                        sd:has_Predicate  alpha:XohmTqLs ;
                        sd:has_Predicate  alpha:OnSBQviK ;
                        sd:has_Predicate  alpha:VahZCMrK .
alpha:XohmTqLs                        sd:num_triples  100000 ;
                        sd:num_objects  79631 .
alpha:OnSBQviK                        sd:num_triples  100000 ;
                        sd:num_objects  86672 .
alpha:VahZCMrK                        sd:num_triples  100000 ;
                        sd:num_objects  96906 .
```

Bibliography

[1] David F. Andersen and Sharon S. Dawes. *Government Information: A Primer and Casebook.* Prentice-Hall, Englewood Cliffs, NJ, 1991.

[2] Lujo Bauer. *Access Control for the Web via Proof-carrying Authorization.* PhD thesis, Princeton University, November 2003.

[3] Lujo Bauer, Scott Garriss, and Michael K. Reiter. Distributed proving in access-control systems. In *Proceedings of the 2005 IEEE Symposium on Security and Privacy*, pages 81–95, May 2005.

[4] Tim Berners-Lee. Notation 3. http://www.w3.org/DesignIssues/Notation3.html, March 2006.

[5] Tim Berners-Lee, James Hendler, and Ora Lasilla. The Semantic Web. *Scientific American*, May 2001.

[6] Abraham Bernstein, Christoph Kiefer, and Marcus Stocker. OptARQ: A SPARQL Optimization Approach based on Triple Pattern Selectivity Estimation. Technical Report ifi-2007.03, Department of Informatics, University of Zurich, March 2007.

[7] Alexandre Bertails and Gautier Poupeau. Lapport des technologies du Web smantique la gestion des donnes structures. http://www.slideshare.net/lespetitescases/lapport-des-technologies-du-web-smantique-la-gestion-des-donnes-structures, 2008.

[8] Dan Brickley and R.V. Guha. RDF Vocabulary Description Language 1.0: RDF Schema. http://www.w3.org/TR/rdf-schema/, February 2004.

[9] K.S. Candan, S. Jajodia, and V.S. Subrahmanian. Secure Mediated Databases. In *Proceedings of the Twelfth International Conference on Data Engineering*, March 1996.

[10] Mathew Cherian, Lalana Kagal, and Eric Prud'hommeaux. Policy Mediation to Enable Collaborative Use of Sensitive Data. In *The Future of the Web for Collaborative Science - World Wide Web Conference*, Raleigh, NC, USA, 2010.

[11] E.F. Codd. A Realtional Model of Data for Large Shared Data Banks. *Communications of the ACM*, June 1970.

[12] Mitchell D. Chester Commissioner. Welcome to the Massachusetts Department of Elementary and Secondary Education. http://www.doe.mass.edu/mailings/welcome.html, 2008.

[13] Robert Bickerton (Associate Commissioner) and Robert Lei (Chief Data Analyst MCAS Group). Interview @ Department of Elementary and Secondary Education, 75 Pleasant St, Malden, MA 02148 on October 13, 2010.

[14] Dave Beazley. SWIG. http://sourceforge.net/apps/mediawiki/swobjects/index.php, 2010.

[15] Sharon S. Dawes. Interagency Information Sharing: Expected Benefits, Manageable Risks. *Journal of Policy Analysis and Management*, Summer 1996.

[16] Decentralized Information Group. AIR Policy Language. http://dig.csail.mit.edu/2009/AIR/, 2009.

[17] Decentralized Information Group and MINDSWAP. Policy-Aware Web Project. http://www.policyawareweb.org/, 2006.

[18] Eric Prud'hommeaux. SWObjects Semantic Web Library. http://sourceforge.net/apps/mediawiki/swobjects/index.php, 2010.

[19] Executive Office for Administration and Finance Mass.Gov. The Benefits of E-Government, 2010.

[20] Decentralized Information Group. Secure Federation Systems for Semantic Web Sources. http://dig.csail.mit.edu/2009/AFOSR/index.html.

[21] Decentralized Information Group. Accountable Information Usage in Fusion Center Information Sharing. http://dig.csail.mit.edu/2009/DHS-fusion/index.html, 2009.

[22] RDF Working Group. Resource Description Framework (RDF). http://www.w3.org/RDF/, February 2004.

[23] J. Keith Harmon and Rae N. Cogar. The Protection of Personal Information in Intergovernmental Data-Sharing Programs, 1998.

[24] David Landsbergen Jr. and George Walken Jr. Realizing the Promise: Government Information Systems and the Fourth Generation of Information Technology. *Public Administration Review*, March/April 2001.

[25] Lalana Kagal, Tim Berners-Lee, Dan Connolly, and Daniel Weitzner. Self-Describing Delegation Networks for the Web. In *IEEE POLICY '06: Proceedings of the Seventh IEEE International Workshop on Policies for Distributed Systems and Networks*, pages 205–214, Washington, DC, USA, 2006. IEEE Computer Society.

[26] Lalana Kagal, Tim Berners-Lee, Dan Connolly, and Daniel J. Weitzner. Using Semantic Web Technologies for Policy Management on the Web. In *21st National Conference on Artificial Intelligence*, 2006.

[27] Lalana Kagal, Chris Hanson, and Daniel Weitzner. Using Dependency Tracking to Provide Explanations for Policy Management. In *IEEE Policy 2008*, 2008.

[28] Dave Kolas. Query Rewriting for Semantic Web Information Integration. In *Sixth International Workshop on Information Integration on the Web*, 2007.

[29] Vladimir Kolovski, Yarden Katz, James Hendler, Daniel Weitzner, and Tim Berners-Lee. Towards a policy-aware web. In *In Semantic Web and Policy Workshop at the 4th International Semantic Web Conference*, 2005.

[30] Andreas Langegger, Martin Blochl, and Wolfram Woss. Sharing Data on the Grid using Ontologies and distributed SPARQL Queries. In *18th International Workshop on Database and Expert Systems Applications*, pages 450–454, Washington, DC, USA, 2007. IEEE Computer Society.

[31] Deborah McGuinnes and Frank van Harmelen. OWL Web Ontology Language. http://www.w3.org/TR/owl-features/, February 2004.

[32] Dennis McLeod and Dennis Heimbimger. A Federated Architecture for Database Systems. In *National Computer Conference*, 1980.

[33] N/A. None. http://www.ibm.com/developerworks/data/library/techarticle/dm-0504zikopoulos/Part2Figure2.gif, April 2005.

[34] N/A. The Data Deluge. *The Economist*, February 25 2010.

[35] N/A. Linked Data - Connect Distributed Data across the Web. http://linkeddata.org/, 2011.

[36] Department of Elementary and Secondary Education. Massachusetts Comprehensive Assessment System: Participation Requirements for Students. http://www.doe.mass.edu/mcas/participation, 2010.

[37] Sharon Wright (Chief Information Officer). Interview @ Executive Office of Education, 1 Ashburton Place, Boston, MA 02108 on July 22, 2010.

[38] Oregon.gov. Oregon E-Government Program. http://www.das.state.or.us/DAS/EISPD/EGOV/benefits.shtml, 2010.

[39] Eric Prud'hommeaux and Andy Seaborne. SPARQL Query Language for RDF W3C Recommendation. http://www.w3.org/TR/rdf-sparql-query, January 2008.

[40] Bastian Quilitz and Ulf Leser. Querying Distributed RDF Data Sources with SPARQL. In *5th European Semantic Web Conference (ESWC2008)*, pages 524–538, June 2008.

[41] Kim Sopko and Nancy Reader. Public and Parent Reporting Requirements: NCLB and IDEA Regulations. http://www.projectforum.org/docs/PublicandParentReportingRequirements-NCLBandIDEARegulations.pdf, 2007.

[42] Tim Berners-Lee. Naming and Addressing: URIs, URLs,... http://www.w3.org/Addressing/, 1993.

[43] Tim Berners-Lee and Vladimir Kolovski and Dan Connolly and James Hendler and Yoseph Scharf. A Reasoner for the Web. http://www.w3.org/2000/10/swap/doc/paper/, October 2000.

[44] Jinpeng Wang, Zhuang Miao, Yafei Zhang, and Jianjiang Lu. Semantic Integration of relational data using SPARQL. In *Second International Symposium on Intelligent Information Technology Application*, pages 422–426, 1996.

[45] Sumudu Watugula. Designing Customizable End User Applications using Semantic Techologies to Improve Information Management. MEng Thesis, MIT, Department of Electrical Engineering and Computer Science, May 2006.