ADDIS ABABA UNIVERSITY

SCHOOL OF GRADUATE STUDIES

FACULTIY OF INFORMATICS

DEPARTMENT OF COMPUTER SCIENCE

Design and Development of Semantic Database Integration System-SeDIS:

(Case study on the Ministry of Health)

A Project paper submitted to the School of Graduate Studies of Addis Ababa University in partial fulfillment of the requirements for the Degree of Masters of Science in Computer Science

By:

Mekonnen Geremew

June 2008

ADDIS ABABA UNIVERSITY

SCHOOL OF GRADUATE STUDIES

FACULTIY OF INFORMATICS

DEPARTMENT OF COMPUTER SCINECE

Design and Development of Semantic Database Integration System-SeDIS:

(Case study on the Ministry of Health)

By

Mekonnen Geremew

Approved by:

1. **Dr. Dejene Ejigu /Advisor/** _____
2. _____ _____

June 2008

ACKNOWLEDGMENT

First of all, my deep and sincerely appreciation goes to my advisor Dr. Dejene Ejgu for his critique, supervision of the project work and friendly approach.

Thanks to Ato Oli and Ato Biniyam from the federal ministry of health for their cooperation to assist me to have a better understanding in the existing system of the federal ministry of health.

My acknowledgment also goes to friends: Emamu Abdela ,Sishu Girma , Dessalegn Adugna and Solomon Hailu who have been providing me the necessary support for the successful completion of this work.

TABLE OF CONTENTS

List of Figures

List of Tables

Abstract

Integration of information from different sources continues to be a high priority in today's business. One of the many sources for integration of information is data stored in relational databases. Database integration is therefore one of the current and burning research issues in the field of information systems.

Ethiopia has started an ambitious program of woreda networking ("woreda-net") to connect Federal, Regional, and Woreda levels and in this perspective a database integration system could provide an integrated source of aggregated and organized data from the regional and woreda levels to the federal offices.

According to our country's structure, the ministry of health is one of the offices at the federal level and has its information sources from regional and woreda health offices. Policy makers at the federal level need these data to be integrated for further processing.

This project work is started by conducting revision of works related to database integration and have identified requirements and challenges of database integration. Different approaches to database integration have also been investigated through review of related works.

In this project, semantic database integration system is proposed to integrate heterogeneous databases.The proposed model distinguishes two major layers: Database to RDF Gateway and, Mapping and Integration Engine. In our model, Database to RDF gateway component converts source databases to RDF/OWL forms by filling their schemas to RDF/OWL template.

To define concepts in the domain of health, we have developed ontology. This ontology will help in resolving semantic conflicts among the component databases. The mapping and integration engine is the main component of our model and it performs the task of mapping and integration. During the mapping process, it relates RDF schemas to concepts on the ontology and during the integration process; it performs the task of reasoning.

Two more components: Data source and application layers are used to test our proposed system. The data source layer consists of candidate databases for integration and the application layer consists of user applications.

Acronyms

The following are some of the abbreviations along with their meaning used in this document.

Acronym/Abbreviation	Meaning
DMS	Data Management system
RDF	Resource Description Framework
OWL	Web ontology language
COBOL	Common Bussiness Oreinted Langauge
DBMS	Database Management Systems
RDBMS	Relational Database Management Systems
OODBMS	Object Oriented Database Management Systems
JDBC	Java Database connectivity
HTML	HyperText Markup Language
GAV	Global-as-view
LAV	Local-as-view
XML	Extensible Markup language
XOL	Ontology Markup language
FMOH	Federal ministry of health
XHTML	Extensible HyperText Markup Language
DTD	Document Type Definition
URI	Uniform Resource identifier
URL	Uniform Resource location
HMIS	Health management system
PPD	Planning and Program Development
ARM	Annual Review Meeting
RDFS	Resource description Framework Schema
API	Application programming interface
DL	Description logics
OGA	Other government organization
NGO	Non government organization

1. Introduction

Integration of information resources continues to be a high priority in businesses today. Companies have tried with the integration of legacy systems for some time, but with the explosion of web-based resources, the interoperability of information has become an even greater problem. The essence of this problem is the implicit and frequently inconsistent semantics of the information. Because the web has made access to information much easier, the potential for companies to integrate information from different sources has grown tremendously. But, for the most part, each information resource was created for a single purpose, and the power of integration is in the merging of information, particularly in unanticipated ways.

Most large organizations maintain their data in many distinct independent databases that have been developed at different times on different platforms and DMS (Data Management Systems). Their new applications will use existing data from various data stores. Hence the new economic challenges force enterprises to integrate their functions and therefore their information systems including databases. Henceforth, the ability to make data stores interoperable becomes a crucial factor for the development of new information systems.

1.1. Problem Statement

The effectiveness of an organization is highly dependent on its use of accurate, timely and well organized information. For most organizations, their data are stored on different heterogeneous databases using different terminologies and even different languages. Ethiopia started an ambitious program of woreda networking (“woreda-net”) to connect

Federal, Regional, and Woreda levels and in this perspective a database integration system could provide an integrated source of aggregated and organized data.

According to our country's structure; the Ministry of Health, which is at a federal level, has its information sources from regional health offices. Policy makers at the federal level need these data to be semantically integrated for further processing. With respect to data management of Ministry of Health, the major problems include:

- Manual integration of data took the Ministry a long duration of time and it is error prone
- Policy makers and other concerned bodies do not get accurate and timely information
- Information sources may contain data of different structures and different semantics .Combing these kinds of data are difficult.

1.2 Objectives

1.2.1 General objective

The main objective of this project is to design and develop a system for semantic integration of databases for the Ministry of Health.

1.2.2 Specific objectives

- Study organization of existing regional health data
- Examine the tools and techniques necessary for database integration
- Propose a database integration model
- Design and develop a semantic database integration system.

1.3 Scope of the Project

The focus of this project is to design and develop semantic database integration that resolves naming conflicts.

1.4 Methodology

To conduct this project, the following methodology was used:

- Different related researches and background theory were revised through literature review.
- Semantically rich languages and tools like RDF and OWL were explored and used.
- To collect data, interviews, questionnaires and document analysis were used
- Sample databases were created
- Sample data were populated to the databases
- The database contents were mapped to the RDF files
- Ontology for the chosen domain was built
- Test applications were built

1.5 Document organization

This document is organized into seven chapters including this. The second and third chapters address, respectively, a review of related works and requirements analysis.

The fourth chapter deals with the design architecture of our proposed system. The fifth chapter describes the implementation of the system and the last two chapters describe conclusions and future works.

2. Literature review and related works

In this review of related works, issues in database integration, problems in database integration, approaches in database integration and works related to each approach will be discussed.

2.1 Database integration

In this sub section, we describe some of the problems involved in achieving interoperability between heterogeneous systems.

2.1.1 Legacy database systems

The presence of legacy data systems is one of the major obstacles in the use of integrated information [1]. Typically, legacy data systems are very large. They are typically written in old programming languages like COBOL. Such systems are usually mission critical and inflexible in nature.

Integrating such systems is very costly because of the complexity of understanding data semantics which is either buried in application programs or was never documented by original designer. The incompleteness of their specifications leads to ambiguities of the interpretation of the data schema. The hardest case is when data resides in files, but understanding unnormalized and poorly documented relational databases also is very difficult [2].

2.1.2 Autonomy

Legacy database systems were typically designed to support local requirements imposed by the local environment, and without considering a possible cooperation with other systems.

In other words, databases are usually under separate and independent control. The different aspects of autonomy are summarized as follows [3]:

1. Design autonomy: The databases have their own data model, query language, semantic interpretation of data, constraints, etc.
2. Communication autonomy: The databases have the ability to decide when and how to respond to requests from other databases.
3. Execution autonomy: The execution order of transaction is controlled by the legacy database. They don't need to inform any other system of the execution order of local or external operations.
4. Association autonomy: The legacy databases are able to decide whether participate or not in one or more federations, as well the possibility of its dissociation of a federation.

2.1.3 Database Heterogeneity

A major obstacle to interoperability of legacy databases is their heterogeneity. Heterogeneity among legacy databases is caused by the design autonomy of their owners in developing such systems. Legacy systems were typically designed to support local requirements, under constraints imposed with a given system.

Databases are heterogeneous on several levels: in their storage methods (flat files and different DBMS) and semantics.

Storage

Many DBMSs for storing data exist. The common once are discussed in this section. **Flatfiles :** Only a few years ago, data was most commonly stored in ASCII text files.

Nowadays, the number of "databases" which are implemented as flat files decreases and many databases are moved from their old flat file representations to DBMSs. Flatfiles are no longer considered to be an appropriate alternative to DBMSs, but rather as a data exchange format. However, flatfiles are still not obsolete, since many applications operate on flat files.

Relational Database Management Systems: Relational databases were firstly introduced in 1970 .Since then, a strong theoretical background and many RDBMS have been developed. Nowadays most databases are implemented on relational DBMS.

Object Oriented Database Management Systems: Although OODBMS are relatively new, they are reasonably well standardized and have a sound theoretical background. Many OODBMS can be accessed via JDBC.

In OODBMS, any complex data types that can be implemented in the object oriented programming language that is used to generate the objects can be stored simply by storing objects.

Semantic Heterogeneity

In computer sciences, the semantics of a programming language describes the relationship between the syntax and the model of computation. Whereas the syntax of programming languages can usually well be formalized, the semantics of programming languages more or less defy formalization. However, with regard to linguistics, the term semantics applies to the meaning of words. In a linguistic sense, semantic conflicts occur for example when the same symbol is used for different things (mouse as computer device or an animal), or when different symbols exist for the same thing .Thus dealing with semantics in a

linguistically sense, involves for example dealing with synonyms or with the disambiguation of the meaning of homonyms.

Such semantic conflicts that are due to inconsistent naming of database tables, attributes or entries also occur between relational databases.

In [4], semantic heterogeneity is described as variations in the manner in which data is specified and structured in different components. Semantic heterogeneity is a natural consequence of the independent creation and evolution of autonomous databases which are tailored to the requirements of the application system they serve.

A more vivid description is given in [5]: "A schema contains a semantic description of the information in a given database. It is possible to define equivalent schemas in as many ways as there are data models. Further, the same (or similar) information can be represented in many ways in the same data model. Given such inter- and intra-model variability, it is indeed a formidable task to integrate many schemas into a homogeneous schema." Semantic conflict can occur at attribute and table levels.

Conflicts in Attributes: At the attribute level, different attributes can have the same name and semantically equivalent attributes may have different names.

In Figure 2.1, the attribute EmployeeId in the table Employee and the attribute IDNo in the table Staff both store employee IDs, but are named differently.

Further conflicts at the attribute level arise when equivalent attributes use different data types. For example a IDNo can either be stored as an integer or a string.

EmployeeId	Emp_First_Name	Emp_Last_Name	address	
Moh/001/98				

Table:Employee

IDNo	FullName	
100		

Table:Staff

Figure 2. 1: Tables showing naming and data type conflicts between attributes

Conflicts in Tables:Similar conflicts can occur between database tables. Databases can contain different tables, which have the same name, or tables that store equivalent data but use different names.

In addition, table structure conflicts arise when similar tables have partly different attributes. For example in Figure 1, the table employee has an attribute address that does not exist in the table staff.

2.2 The Integration Problems

The integration problem refers to the problem associated with integrating the data from two or more different data sources. Integration is often required between applications or databases with widely differing views on the data and how it is organized. Thus, integration is hard because conflicts at both the structural and semantic level must be addressed. Further complicating the problem is that most systems do not explicitly capture semantic

information. This forces designers performing the integration to impose assumptions on the data and manually integrate various data sources based on those assumptions.

The two basic levels of integration are schema-level integration and data-level integration. The higher-level of integration is at the schema level. The schema contains the format, structure, and organization of the data in a system. Most systems do not capture operational or data semantics at the schema level. Schema integration is the process of combining database schemas into a coherent, global view. After schema integration has been completed, there is the related problem of data integration. Data integration focuses on integrating information at the data item level.

2.2.1 The Schema Integration Problem

Schema integration is the process of combining database schemas into a coherent, global view. Since database systems tend to be developed in isolation, it is often hard to combine different database schemas because of different data models or structural differences in how the data is represented and stored [6]. It is difficult to determine when two schemas represent the same data in different databases, even if they were developed under the same data model. Each database system has its own world view, and the task of integrating these "views" is as difficult as integrating the knowledge of two humans [7]. Thus, there are many factors which may cause schema diversity [7]:

- Different user or view perspectives
- Equivalence among constructs of the model (eg. a designer uses an attribute in one schema and an entity in another)

- Incompatible design specifications (eg. cardinality constraints, bad name choices)
- Common concepts can be represented by different representations which may be identical, equivalent, compatible, or incompatible
- Concepts may be related by some semantic property which arises only by combining the schemas (inter schema properties)

There are several features of schema integration which make it difficult. The first problem is that data may be represented using different data models. For example, one database may use the relational model, while another database may use an object-oriented model. Even when two databases use the same data model, both naming and structural conflicts may arise.

Naming conflicts occur when the same data is stored in multiple databases, but it is referred to by different names. Naming conflicts arise when names are homonyms and when names are synonyms. The homonym naming problem is when the same name is used for two different concepts. The synonym naming problem occurs when the same concept is described using two or more different names.

Structural conflicts arise when data is organized using different model constructs or integrity constraints. Some common structural conflicts [7] are:

- Type conflicts - using different model constructs to represent the same data
- Dependency conflicts - group of concepts related differently in different schemas (eg. 1-to-1 participation versus 1-to-N participation)
- Key conflicts - different keys for the same entity

A robust integration methodology must be able to handle both naming and structural conflicts.

2.2.2 The Data Integration Problem

Data integration is the process of combining data at the entity-level. After schema integration has been completed, a uniform global view has been constructed. However, it may be difficult to combine all the data instances in the combined schemas in a meaningful way. Combining the data instances is the focus of data integration.

Data integration is difficult because similar data entities in different databases may not have the same key. Entity identification [7] is the process of determining the correspondence between object instances from more than one database. Combining data instances involves entity identification (determining which instances are the same), and resolving attribute value conflicts (two different attribute values for the same attribute).

Data integration is further complicated because attribute values in different databases may disagree or be range values.

2.3 Approaches to database integration

Different approaches to database integration are discussed and reviewed in [8]. In [9] four approaches for database integration are discriminated. These are:

• Hypertext Navigation, i.e. the HTML front ends of Databases are interlinked.

• Data Warehouse, i.e. physically merging (converting, importing...) of several databases into one big database.

• Multi Database Queries, i.e. querying several databases at the same time.

• Federated Databases. In contrast to “Multi Database Queries”, federated databases integrate database schemata in a federation layer, although like in multi database approaches each database remains autonomous.

2.3.1 Hypertext Navigation Systems

At present, most databases are connected to the Internet and can be accessed via web pages. Many databases provide hypertext links to entries in other databases.

2.3.2 Database Mediation and Federation

Database integration systems which use mediation or federation typically consist of three elements: wrappers (W1,W2,W3), an integration layer and a query interface (see Figure 2.2). The wrappers provide uniform access to the heterogeneous data sources. The integration layer decomposes user queries, sends them to the relevant wrappers and finally integrates the query results before the result is returned to the user via the query interface. In addition, often several other components such as administrative tools and query optimizer exist.

In contrast to mediated databases, in federated databases wrappers mainly map the different interfaces between the data sources. Thus in database federations it is required that the data sources provide the main search and query functionality via different interfaces so that the wrappers mainly have to translate between the different interfaces.

In many cases the data sources do not provide suitable search or query methods. Examples are web pages, flat files and other more or less unstructured data sources. In mediated databases, as the term mediation emphasizes, the wrappers play a more active role and implement when necessary missing search or query methods for the data sources.

Mediator based approach has two further approaches; Global-as-view (GAV) and Local-as-view (LAV) [10]. In GAV approach, mediator is based on the source relation schema and it facilitates great query reformulation .While in LAV approach, source relation is based on the mediator's schema and relation. GAV approach has its drawback in adding or removing sources as it will involve modification of the mediator schema. Unlike GAV, in LAV, adding or removing sources is much simpler. However LAV makes the query reformulation complicated.

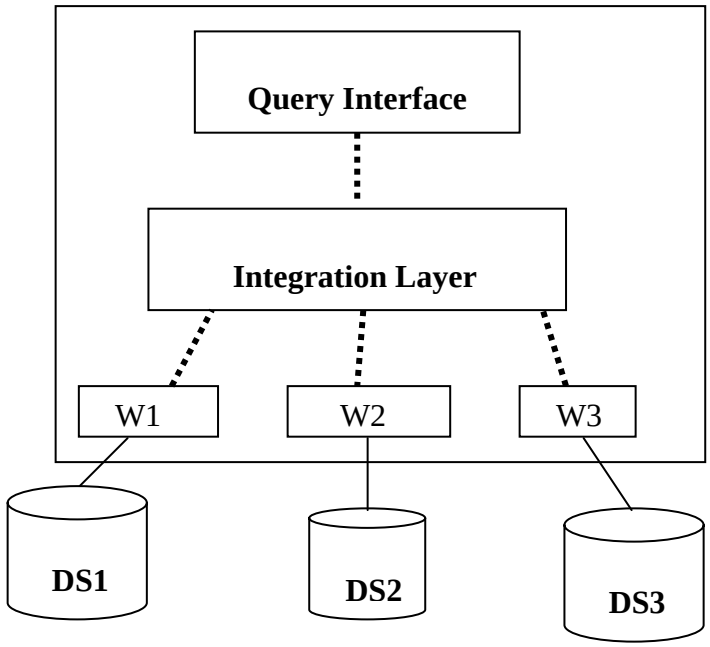

Figure 2. 2 : Typical architecture of database mediation and federation systems.

Mediator based approach has the advantage that it would not have the update problem as the query directly goes to the original source. Mediator based can be looked as a cheaper and effective approach since it involves schema or view integration, rather than to have huge storage to store copied data from all the involved data sources.

While the federated database approach ensures data is concurrent or synchronized and is easier to maintain (when new databases are added), it generally has a poorer query performance than the warehouse integration approach.

2.3.3 Data Warehouses

The data warehouse approach involves translating data from different sources into a local data warehouse, and executing all queries on the warehouse rather than on the distributed sources of that data. This approach eliminates various problems including network bottlenecks, slow response times, and the occasional unavailability of sources. In addition, creating a warehouse allows for an improved query efficiency or optimization since it can be performed locally [11]. The approach is also reliable because there are fewer dependencies on network connectivity. Another benefit in this approach is that it allows values (e.g., filtering, validation, correction, and annotation) to be added to the data collected from individual sources. The most important advantage of using the data warehousing approach is, since underlying data sources may contains errors, it keeps a separate copy of data called cleansed copy.

Data warehouse approach does have its tradeoffs. Results reliability and overall system maintenance are questionable as there are possibilities of returning outdated results.

Changes in data sources do not mean the data in the warehouse will also be changed, so there is a need of detecting changes in data sources as well as automating the update of the data warehouse. The warehouse needs to be periodically updated to reflect the modifications of the source databases.

3. Semantic Database Integration

The information society demands complete access to available information, which is often heterogeneous and distributed. In order to establish efficient information sharing, many technical problems have to be solved. First, a suitable information source must be located that might contain data needed for a given task. Once the information source has been found, access to the data therein has to be provided. This means that each of the information sources found to work together with the system that is querying the information. The problem of bringing together heterogonous and distributed computer systems is known as interoperability problem.

Interoperability has to be provided on a technical and on an informational level. In short, information sharing not only needs to provide full accessibility to the data, it also requires that the accessed data may be processed and interpreted by the remote system. One of the heterogeneity problems is semantic heterogeneity. Semantic heterogeneity considers the contents of an information item and its intended meaning.

In order to achieve semantic interoperability in a heterogeneous information system, the meaning of the information that is interchanged has to be understood across the systems.

Semantic conflicts occur whenever two contexts do not use the same interpretation of the information. Goh identifies three main causes for semantic heterogeneity [12]:

- Confounding conflicts occur when information items seem to have the same meaning, but differ in reality, e.g. owing to different temporal contexts.
- Scaling conflicts occur when different reference systems are used to measure a value. Examples are different currencies.

- Naming conflicts occur when naming schemes of information differ significantly. A frequent phenomenon is the presence of homonyms and synonyms.

Researchers in the fields of databases and information integration have produced a large body of research to facilitate interoperability between different systems. This research ranges from techniques for matching database schemas to answering queries using multiple sources of data. Ontology research is another discipline that deals with semantic heterogeneity in structured data.

The use of ontologies for the explication of implicit and hidden knowledge is a possible approach to overcome the problem of semantic heterogeneity. Uschold and Gr¨uninger [13] mentioned interoperability as a key application of ontologies, and many ontology-based approaches [13] to information integration in order to achieve interoperability have been developed.

3.1 Ontology

The most commonly quoted definition of ontology is “a formal, explicit specification of a shared conceptualization” [13]. A conceptualization, in this context, refers to an abstract model of how people think about things in the world, usually restricted to a particular subject area. An explicit specification means that the concepts and relationships in the abstract model are given explicit names and definitions. The name is a term, and the definition is a specification of the meaning of the concept or relation. A definition says how a term necessarily relates to other terms. Formal means that the meaning specification is encoded in a language whose formal properties are well understood—in practice, this

usually means logic-based languages that have emerged from the knowledge representation community within the field of Artificial Intelligence. Formality is an important way to remove ambiguity that is prevalent in natural language and other informal notations; it also opens the door for automated inference to derive new information from the meaning specifications. Shared means that the main purpose of ontology is generally to be used and reused across different applications and communities.

There has been much discussion on what exactly counts as an 'ontology'; however there is a common core that runs through virtually all approaches:

- a vocabulary of terms that refer to the things of interest in a given domain;
- some specification of meaning for the terms ideally grounded in some form of logic.

Ontologies represent many different kinds of things in a given subject area. These things are represented in the ontology as *classes* (sometimes called *concepts*) and are typically arranged in a lattice or taxonomy of classes and subclasses. Each class is typically associated with various *properties* (also called *slots* or *roles*) describing its features and attributes as well as various restrictions on them (sometimes called *facets* or *role restrictions*). An ontology together with a set of concrete *instances* (also called *individuals*) of the class constitutes a *knowledge base*. The lattice or taxonomy of classes is a primary focus of most ontologies.

3.2 Ontologies vs. Database Schema

There are many interesting relationships between database schema and formal ontologies. We will consider the following issues: language expressivity, systems that implement the languages and usage scenarios.

There is much overlap in expressivity, including: objects, properties, aggregation, generalization, set-valued properties, and constraints. For example, entities in an ER model correspond to concepts or classes in ontologies, and attributes and relations in an ER model correspond to relations or properties in most ontology languages. For both, there is a vocabulary of terms with natural language definitions. Such definitions are in separate data dictionaries for DB schema, and are inline comments in ontologies. Arguably, there is little or no obvious essential difference between a language used for building DB schema and one for building ontologies. There are many specific differences in expressivity, which vary in importance. Many of the differences are attributable to the historically different ways that DB schema and ontologies have been used. Ontologies have a range of purposes including interoperability, search, and software specification. One or more parties commit to using the terms from the ontology with their declared meaning. The primary use of most DB schema is to structure a set of instances for querying a single database. This difference impacts heavily on the role of constraints.

For ontologies, constraints are called axioms. Their main purpose is to express machine-readable meaning to support accurate automated reasoning. This reasoning can also be used to ensure integrity of instances in a knowledge base. For databases, the primary purpose of constraints is to ensure the integrity of the data (i.e. instances). These ‘integrity constraints’ can also be used to optimize queries and help humans infer the meaning of the terms. Cardinality and delete constraints are important kinds of integrity constraints which have highly DB specific uses that are outside the scope of most or all ontology systems. For example, cardinality constraints are used for getting the foreign key in the right direction and to ensure that extra tables are built for many to many relationships. Such constraints do

express meaning, but this may be of secondary importance. The main role for cardinality constraints in ontologies is to express meaning, and ensure consistency (either of the ontology, or of instances). There are some key similarities and differences in systems that implement DB schema languages and ontology languages. For both, there are processing engines that can be used to perform reasoning. SQL engines are highly specialized and tuned for answering queries, reasoning with views and ensuring data integrity. They can handle rich and expressive logical expressions, as can ontology engines. An important difference is that reasoning over ontologies normally is done by general logic-based theorem provers, specific to the language. Although ontology inference may be used for queries and to ensure integrity of instances, these are optional. The fundamental role of a reasoning engine is to derive new information via automated inference. Inference can also be used to ensure the logical consistency of the ontology itself. Note that deriving new information and checking consistency can take place with or without instances. Classically, such mixing of types with instances does not take place with DB schema and data. This is mainly due to much greater scale and performance requirements for database systems. Another key difference is support for taxonomic reasoning: it is fundamental for nearly all ontology applications, but it is not supported by most DBMS.

3.3 Using Ontologies for Database Integration

Initially, ontologies are introduced as an”explicit specification of a conceptualization” [13]. Ontologies can on the one hand be used to define a common controlled vocabulary and on the other hand to semantically define databases. Controlled vocabularies are named lists of terms that are well defined and may have an identifier. The elements of a controlled vocabulary are called concepts. Concepts can either be defined implicitly or by explicitly

listing them. Therefore, ontologies can be used in an integration task to describe the semantics of the information sources and to make the contents explicit .With respect to the integration of sources; they can be used for the identification and association of semantically corresponding information concepts.

3.4 The Role of Ontologies

Content Explication: In nearly all ontology–based integration approaches ontologies are used for the explicit description of the information source semantics. But there are different ways of how to employ the ontologies. In general, three different directions can be identified: single ontology approaches, multiple ontologies approaches and hybrid approaches [14].

Single Ontology approaches: Single ontology approaches use one global ontology providing a shared vocabulary for the specification of the semantics. All information sources are related to one global ontology. Figure 3.1 shows single ontology for content explication.

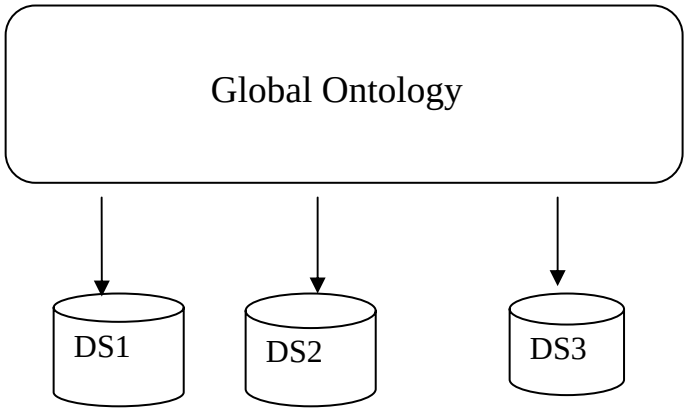

Figure 3. 1 : Single Ontology Approach

Multiple Ontologies In multiple ontology approaches, each information source is described by its own ontology (Figure 3.2).

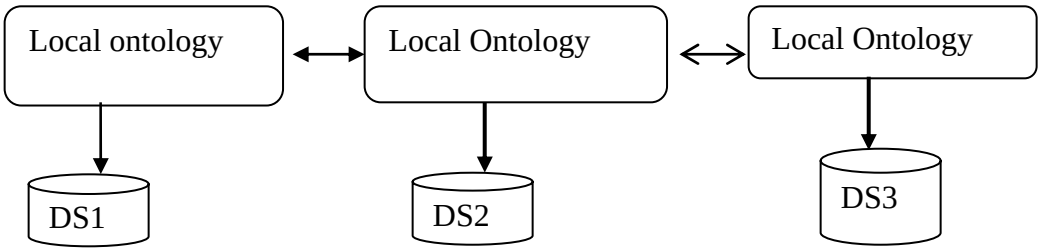

Figure 3. 2: Multiple ontology approach

Hybrid Approaches: Similar to multiple ontology approaches the semantics of each source is described by its own ontology. But in order to make the source ontologies comparable to each other they are built upon one global shared vocabulary [14]. The shared vocabulary contains basic terms (the primitives) of a domain (Figure 3.3).

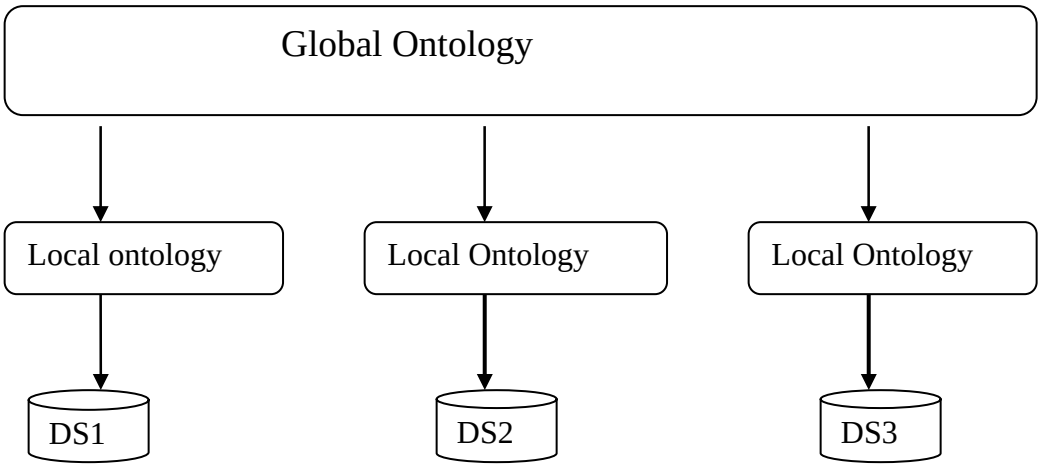

Figure 3. 3 : hybrid ontology approach

Query Model: Integrated information sources normally provide an integrated global view. Some integration approaches use the ontology as the global query schema.

Verification: During the integration process several mappings must be specified from a global schema to the local source schema. The correctness of such mappings can be considered ably improved if these can be verified automatically.

3 .4 Ontology Languages

Several ontology languages have been developed during the last few years .Some of them are based on XML syntax ,such as Ontology Exchange Language (XOL), SHOE (which was previously based on HTML), and Ontology Markup Language (OML), whereas Resource Description Framework (RDF) and RDF Schema are languages created by World Wide Web Consortium (W3C) working groups. Initiatives from DARPA and European IST project are DAML [15] and OIL [16] .Use of DL and automatic classification of concepts through the use of inferencing are the main features of OIL and DAML that further the quest of knowledge representation formalism on the web. Joint committee of the US and EU lay ground later for DAML + OIL. Considering the strength of RDF, OIL and DAML w3c adopted a standard ontology language for the web- the Web Ontology Language (OWL) [17].

XML

While the HyperText Markup Language (HTML) is used for providing a human-friendly data display, it is not machine-friendly. In other words, computer applications do not know the meaning of the data when parsing the HTML tags, since they only indicate how data should be displayed. To address this problem, the extensible Markup Language (XML) was introduced, to associate meaningful tags with data values.

XML adds tags to a text stream to provide some structure and additional information in the same way that HTML does. In addition, a hierarchical (element/sub-element) structure can be created using these tags .With such descriptive and hierarchically-structured labels, computer applications are given better semantic information to parse data in a meaningful way. Despite its machine readability, as indicated by Wang et al. [18], the nature of XML is syntactic and document-centric. This limits its ability to achieve the level of semantic interoperability required by the highly dynamic and integrated applications.

The introduction of the Semantic Web [19] has taken the usage of XML to a new level of ontology-based standardization. In the Semantic Web realm, XML is used as an ontological language to implement machine-readable ontologies built upon standard knowledge representation techniques.

XHTML (Extensible HyperText Markup Language) was delivered in January 2000 by W3C to bring advantages of XML directly to HTML. This language, built on XML, was expected to improve accessibility and functionality on the web. There are not many new or deprecated tags and attributes in XHTML compared to HTML; there is mainly just a new set of coding rules. XHTML was also easily extensible by any other XML documents, which means, for instance, that it could embed fragments of RDF.

Such a standardized grammar for XML documents can be formally defined by Document

Type Definition (DTD) or XML Schema .The older way is DTD, which specifies valid elements and their attributes. XML Schema is a more recent language for restricting the structure of XML documents and also extends XML with data types. Moreover, it uses XML as serialization syntax.

An XML document that has an associated grammar and conforms to the rules defined by this grammar is said to be valid. However, these features were never properly supported by major web browsers and thus it seems the web is not yet prepared for the technology.

However, XML and its grammar languages provide only syntax for structured document and have no capability to express its exact meaning. This drawback makes it difficult to integrate or interchange XML documents automatically. To express a semantic of information the Resource Description Framework (RDF) was developed.

RDF

To address XML semantic limitation the Resource Description Framework (RDF) was developed as a language capable of expressing meaning of information.

Compared to XML, which is document-oriented, RDF takes into consideration a knowledge oriented approach that is designed specifically for the web. One of the advantages of RDF over XML is that an RDF graph depicts in a unique form the information to be conveyed while there are several possibilities to represent the same semantic graph in XML .

RDF, based on the concept of semantic networks, provides a simple yet powerful data model for semantic assertions. This model is based on so called triples. Using these triples a semantic of information is formulated like an elementary sentence consisting of a subject, verb and object.

The subject and object are nodes of the graph, and the predicate is an arc (see Figure 3.4). The subject is a resource being described by the statement, the predicate is a specific property of the subject and the object is a value of the property. Resources (subjects) and

predicates (verbs) are identified by URI, and a value of a property can either be literal or be another URI identified resource.

In the semantic web, RDF is used to describe resources in a machine-understandable way. RDF metadata can contain assertions about anything that can be identified by URI. While the most common URI schema is web URL for web resources identification, assertions can also be made about real world entities such as books or even people.

There are various serialization syntaxes for RDF, but the most common one for a RDF document exchange is RDF/XML (based on XML), since RDF was designed to be complementary to XML in order to specify semantics for data based on XML in a standardized and interoperable manner.

However, RDF is only a tool for resource description; it is not able to define concepts

in a knowledge domain. Ontologies are used to specify a vocabulary of terms and their relationships for RDF documents in the semantic web.

An RDF document can be serialized using several formats. The most common are RDF/XML N3 ,NTRIPLES , TRIX , and Turtle . The RDF/XML is Serialization syntax based on XML.

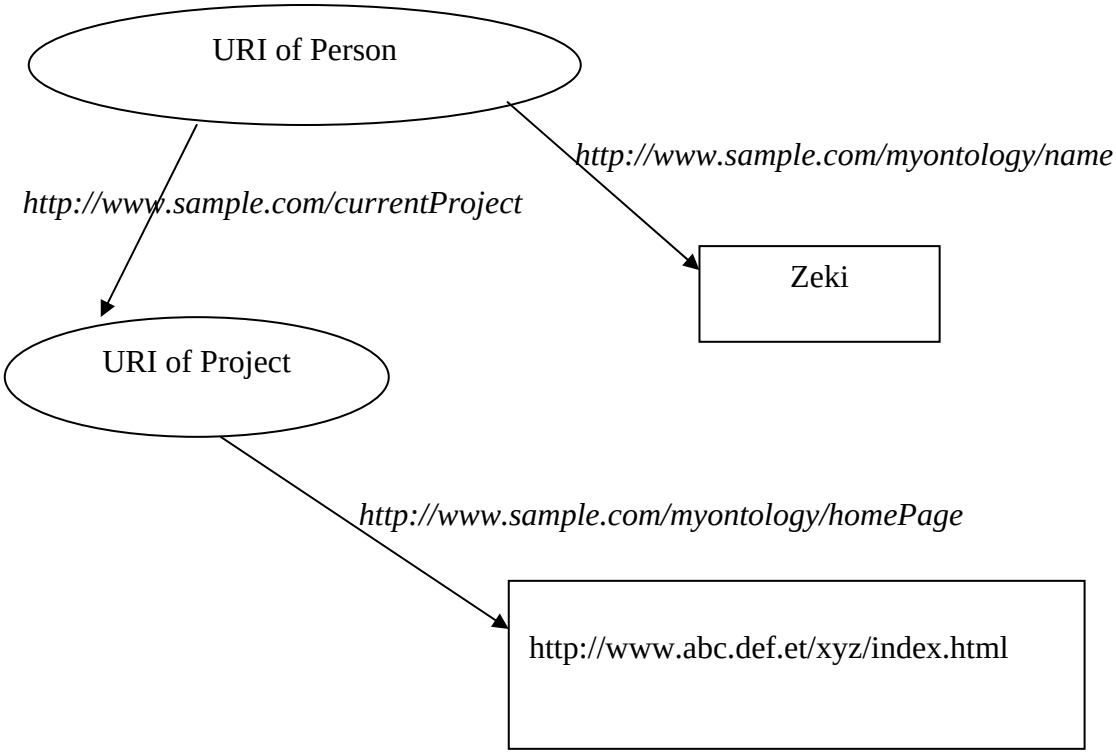

Figure 3. 4: RDF Graph Example

OWL

While RDF and RDFS are commonly-used Semantic Web standards, neither is expressive enough to support formal knowledge representation that is intended for processing by computers. Such a representation consists of explicit objects. Representing knowledge in such explicit form enables computers to draw conclusions from knowledge already encoded in the machine-readable form. More sophisticated XML-based knowledge representation languages such as the Web Ontology Language [20] have been developed. OWL is based on description logics (DL) [21], which are a family of class-based (concept-based) knowledge representation formalisms [21]. They are characterized by the use of various constructors to build complex classes from simpler ones, an emphasis on the decide-ability of key reasoning problems, and by the provision of sound, complete and (empirically)

tractable reasoning services. Description Logics, and insights from DL research, had a strong influence on the design of OWL, particularly on the formalization of the semantics, the choice of language constructors, and the integration of data types and data values.

4. Requirement Analysis

4.1 Existing System

The current system was designed to respond to a top-down, command and control environment, in which decisions were made centrally by specialists. Peripheral health workers executed the decisions made at the center and had little authority or responsibility to make decisions about priorities or resource allocation. In such a setting the information system's job is to move data to the specialists at the center as efficiently as possible.

The current health management system (HMIS) has the strong vertical data flows of a classic command and control system. In general at the facility level, data are recorded daily, as services are provided, on medical records, registers, and tally sheets. Each month these data are compiled and flow from community and facilities to woredas. Data flow quarterly to the zone or sub-city, then to the regional level (see figure 3.1). In general, information stops at each level for processing. Data are almost reported on multiple forms. Specialist, vertically oriented programs at the federal ministry of health (FMOH) receive data quarterly or biannually. Throughout the system, it is not uncommon to have missing data and late reports.

The vertical programs' offices at the federal level are one node in a reporting network that stretches from the most peripheral facility to the FMOH; these offices receive their data on prescribed form at prescribed times. There is an annual reporting mechanism in which planning and program department's (PPD) HMIS unit collects information from all programs and regions for the Annual Review Meeting (ARM) and for PPD's yearly publication of Health and Health-related Indicators. This is the only information that flows directly to the HMIS unit and this flow is initiated by a request from PPD.

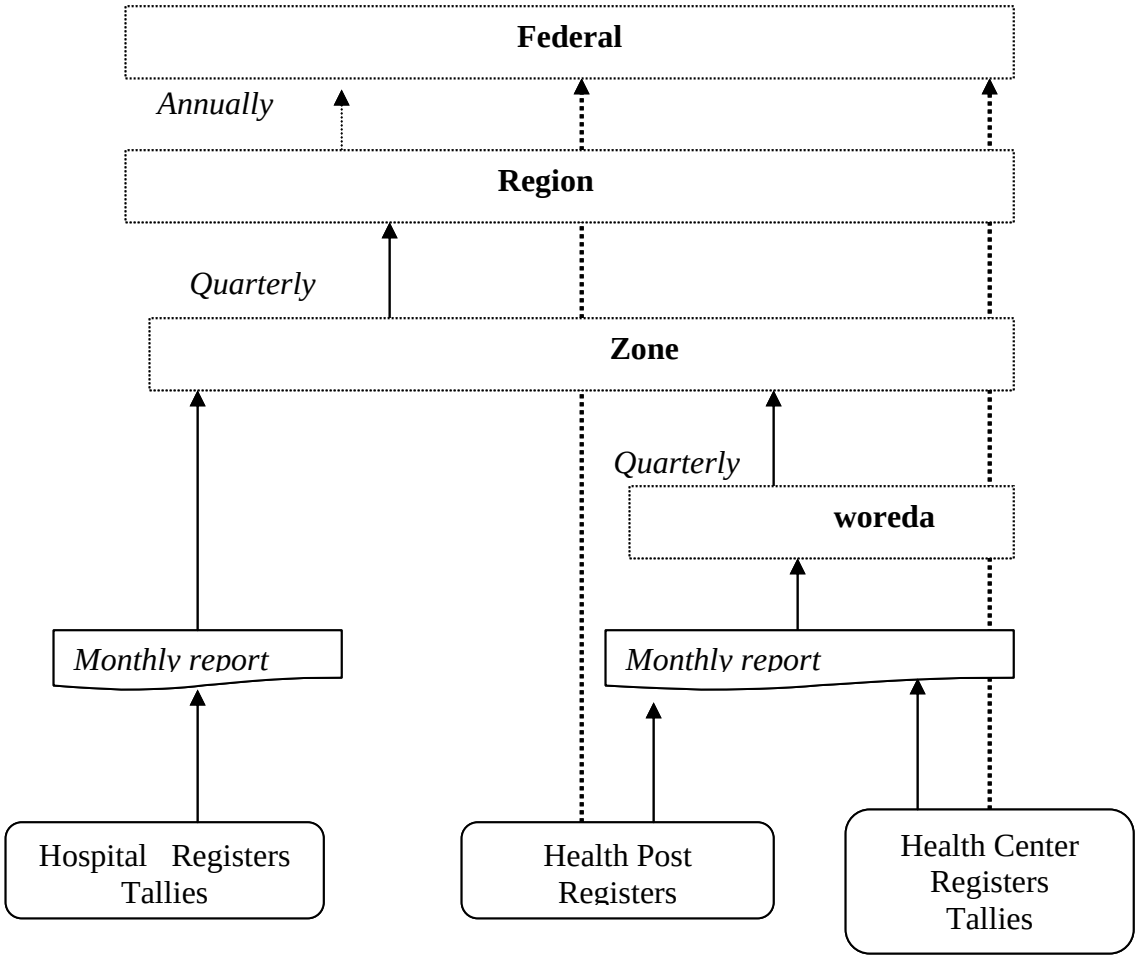

Figure 4. 1: Information flow in current system

There are regional variations from the data flow presented in Figure 4.1.Hospitals usually bypass the woreda level .Surveillance programs (such as Integrated Disease Surveillance and Response (IDSR), tuberculosis, leprosy, and malaria) may forward information directly to zones /sub cities , or to regions at the same time they notify the woreda. Health Post information may go to the supporting Health center and woreda levels. These inconsistencies can make data from different locations incomparable.

The parallel reporting systems may also contain duplicate information. For example, malaria cases are typically reported on at least three different forms that make their way to three different offices. This duplication creates an unnecessary data burden on reporting

facilities that fill the forms. It may also produce inconsistent results. Tables 4.1, 4.2, 4.3 and 4.4 show work flow at national, regional/zonal woreda and facility levels.

Activities at different levels

Table 4. 1: Federal Level Detailed activities

ctivity	Reason	Method
Acquire data	As input for distribution of Information at the national level As a requirement of getting the data from the regional level. To evaluate performance	By letter writing Telephone/Fax E-mail Personal visit
Compile/ Aggregate	To see the national picture	Using computer
Analyze the data	To convert the data into information	Using computer
Prepare the bulletin	For planning, monitoring and evaluation at all levels	Submitted to printing press
Distribution	To reach stakeholder For check-up at the latter stage	Hardcopy distribution through post office Collecting through individual
Repository	For the need of latter time	Keeping hardcopy using folder

Table 4. 2: Regional/Zonal Level Detailed Activities

Activity	Reason	Method
Acquire	To monitor the lower level Achievement To evaluate performance	By letter writing Telephone e-mail personal visit
Compile/aggregate	To see the regional picture	Using Computer
Organize as per the report Format	To respond to the receiver To take action	Using computer

Table 4.3:Woreda Level Detailed Activities

Activity	Reason	Method
Acquire data	To monitor the lower level achievement, to evaluate the performance & for supervision	Hand delivery
Compile/Aggregate	For reporting	Manual
Transmit the data	For the consumption higher level ,for follow up	Hand delivery Hard copy distribution

Table 4. 4: Health Facility Level Detailed Activities

Activity	Reason	Method
Produce/collect the data	For recording the diagnosis/cases, for reporting	Manual
Compile/aggregate	For reporting, for follow-up of cases, for monitoring of epidemics	Manual
Transmit	To respond to the higher level, for planning, for follow-up, for monitoring & evaluation	Manual

4.2 Proposed System

As explained in the existing system analysis, most of the activities undertaken at different levels of health offices are recording data, filling forms, tallying sheets, compiling /aggregating data and passing reports to the next health offices according to the hierarchy until they reach FMOH.

In performing the above activities by the existing system, the following problems are identified.

- Manual integration of data took the Ministry a long duration of time and it is error prone
- Policy makers and other concerned bodies do not get accurate and timely information

- Information sources may contain data of different structures and different semantics .Combing these kinds of data are difficult.
- The processes have many non-value added steps, some of which are required and some used to handle breakdowns in performance.
- Hundred of work steps are currently part of the flow of data gathering and information use, beginning with the meeting of health providers with patients or clients and continuing with registering, tallying compiling, transporting and analyzing
- The system imposes an excessive data burden on health workers. The data burden has been investigated from different perspectives including: the number of filled forms, time spent to fill the forms, and data items to be filled(the number of fields on the form) .The degree of burden varies from one region to another since the number of forms and data items to be filled varies from region to region.
- The system imposes the same data item to be collected on different forms resulting on data duplication. This duplication creates a needless data burden on the health worker and often leads to inconsistent data.

4.2.1 Functional Requirements

To alleviate some problems of the existing system, the new system should provide the following functionalities:

- Enable system administrator of FMOH to combine regional databases
- Enable a user to write a query on the combined databases.

4.2.2 Non-functional Requirements

The system should provide the following non-functional requirements.

- It should be able to transparently integrate different databases
- It should be able process user requests in small amount of time
- It should be easy to use

5 System Design

In this section, the design goals, components that are necessary to realize semantic database integration and the design architecture will be described.

5.1 Design Goals

Design goals are used to identify the expected qualities of the system. In designing database integration systems, several issues must be carefully considered. Issues that we considered as design goals are:

- **Transparency**: The system must allow applications to transparently view and query the different data sources as one uniform data source
- **Extensibility**: The system must support modification of the source databases to provided new functionalities.
- **Usability**: the system should be designed in such a way that users should find convenient to interact with it.

5.2 Architecture

A software system is a set of communicating entities that collaborate to perform a task. The architectural design shows these entities, their relationships and the relationship to the

actors in the system. Figure 5.1 depicts the architecture we used to achieve semantic database integration of databases and helps in meeting these requirements. It consists of components which are grouped in four levels:

- Level 1:Data sources
- Level 2: Database to RDF Gateway
- Level 3:Mapping and Integration Engine
- Level 4: Application

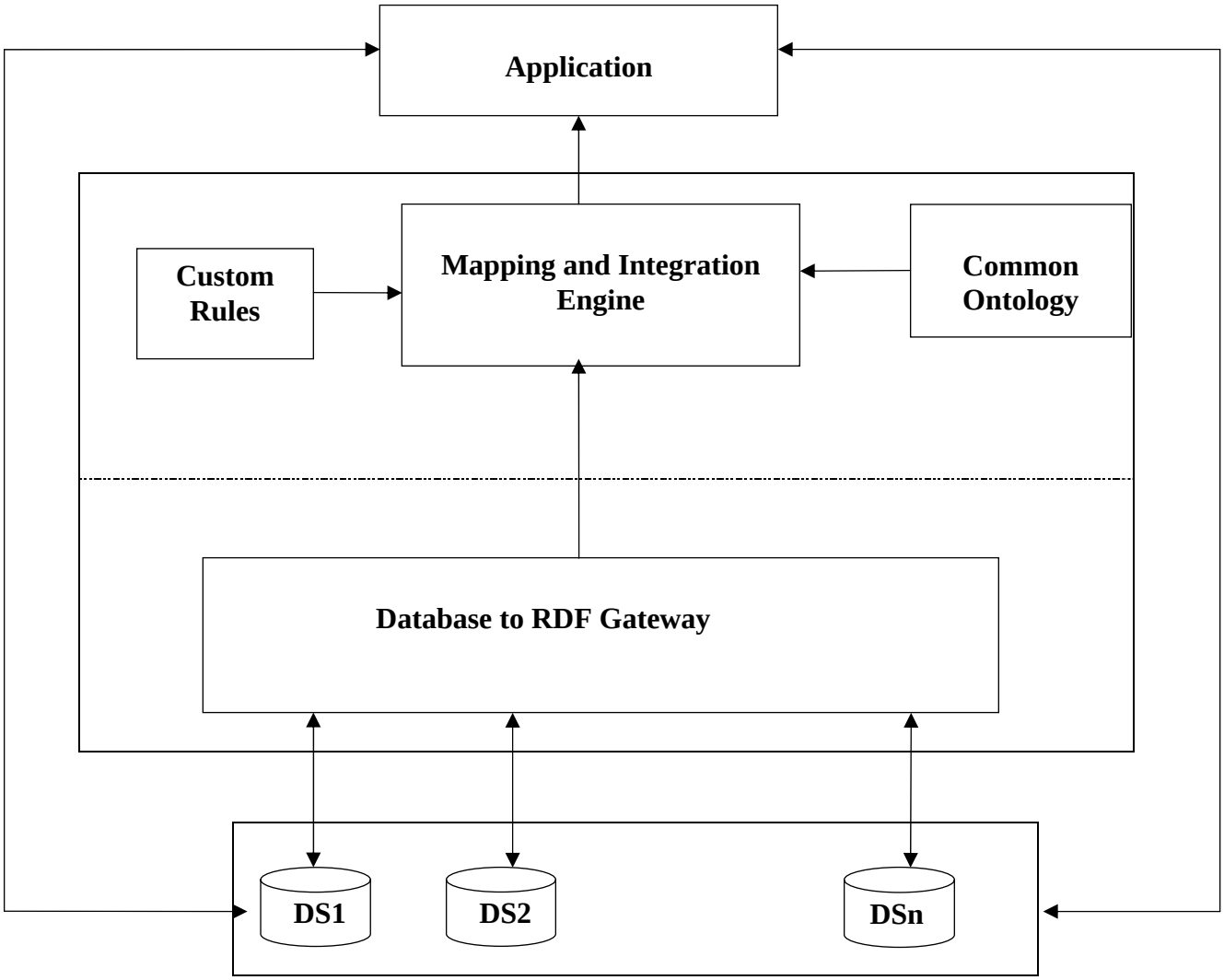

Figure 5.1: Integration Architecture

5.2.1 Data Sources

Level 1 or the data source level consists of information provider's data and schema. The sources may use different relational database management systems for storing and managing their data. These sources are autonomous in all aspects and they can be modified at any time based on the need of their owners.

5.2.2 Database to RDF Gateway

Level 2 or the database to RDF Gateway is responsible for creating RDF/OWL files from the constituent databases. These files will then be combined together to give one uniform RDF file so that it will be passed to ontology layer as an input for creating a mapping file.

Several tools were developed to convert relational database data to ontology data. Some of these tools include: D2R MAP [24], D2R [23], R2O [22],RDQuery [25],and D2RQ[26].For our purpose, we adopted and implemented the algorithm presented in [27].The overall idea of the algorithm is to map sql query to RDF/OWL XML template. Figure 5.2 shows the components of the algorithm.

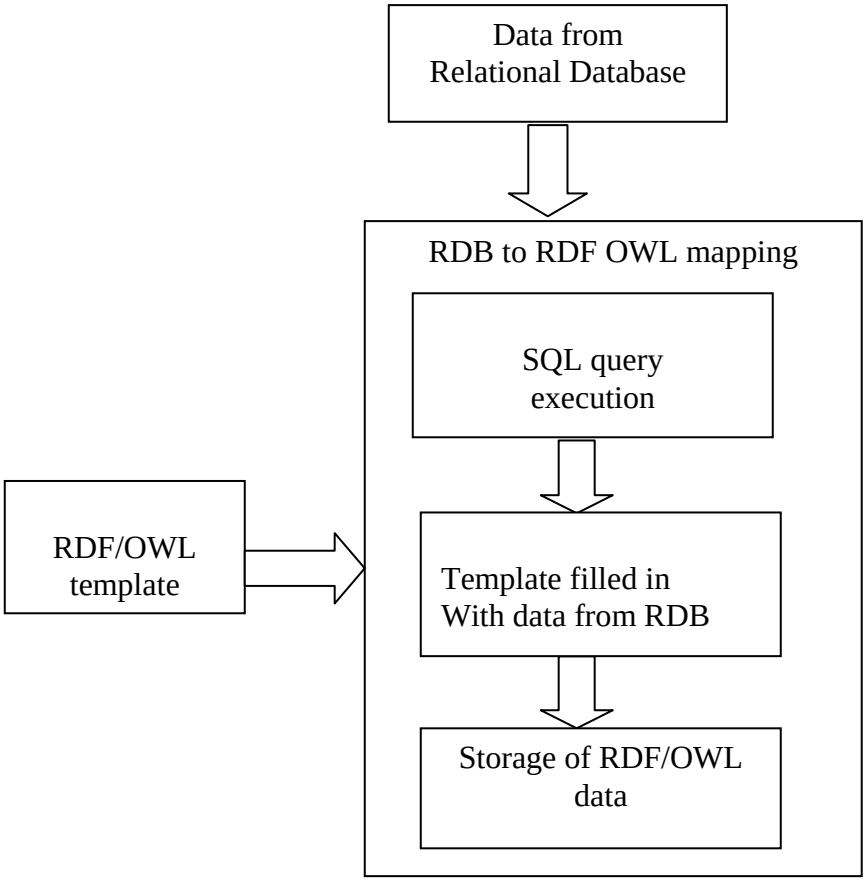

Figure 5. 2 : Components of the algorithm that maps sql to RDF/OWL template

The first component of the algorithm is database tables which contain the data and schemas to be filled in the RDF/OWL template. In the second component, sql statement that retrieves part or all of the content of table data and schemas is executed. In the third component, the result of the execution is filled in the RDF/OWL template. Figure 5.3 shows the RDF/OWL template we used to fill the database schemas and the database data. In the last component, the filled RDF/OWL template is saved on disc for later use.

<?xml version="1.0"?

<**rdf:RDF**

xmlns:rdf=’http://www.w3.org/1999/02/22-rdf-syntax-ns#’

xmlns:addis='http://addisho/fmho/resource/addis#’

xmlns: prefix 'http://prefix ho/fmho/resource/prefix#’

```
        xmlns:owl="http://www.w3.org/2002/07/owl#";
    >
            while(not end of rows of the record set){
                <rdf:Description rdf:about="http://prefix
                ho/fmho/resource/prefixworeda/getObject(1)>
                <rdf:type>
                    <owl:Class rdf:about="http://prefix ho/fmho/resource/prefix/>
                </rdf:type>
                    i=1
                while(not end of column of a record)
                 <prefix getColumnName(i)>getObject(i)
                  </prefix  getColumnName(i)>
                        i++
            </rdf:Description>
  }
</rdf:RDF>
```

Figure 5. 3 : RDF/OWL template

5.2.3 Mapping and Integration Engine

In this level, there are three sub components: common ontology, custom rules and mapping and integration engine.

Common Ontology

Ontologies are introduced as an”explicit specification of a conceptualization”. Ontologies on one hand can be used to define a common controlled vocabulary and on the other hand to semantically define databases.

In our architecture, common ontology sub component contains definitions and semantics of concepts of the chosen domain and it will be used by the mapping and integration engine.

Custom rules

These are additional constraints or rules that the database source schemas should satisfy during the integration process and it is used by the mapping and integration engine.

Mapping and Integration Engine

The mapping and integration engine maps the database source schemas to the ontology schemas based on the semantic relationships between schemas of the database sources. It also integrates the databases by using the custom rules and the mapping schema. We used Jena reasoner as an integration engine. The integration engine performs query translation and data translation.

During query translation, the integration engine extracts data expressed using one schema to answer the query in another schema and during data translation it translates data from a source schema to a target schema.

Jena is a Java framework for semantic Web applications and it is an open source grown out of work with the HP Labs Semantic Web program. It provides a programming environment for RDF, RDFS, and SPARQL and includes a rule based inference engine. The Jena Framework includes a RDF API, reading and writing in RDF/XML, an OWL API, in-memory and persistent storage and SPARQL query engine. Jena has classes to represent graphs, resources, properties and literals. In Jena, a graph is called a model and is represented by the Model interface.

Jena provides a consistent programming interface for ontology application development, independent of the type of ontology language used in programs. That is, the Jena ontology API is language-neutral. Jena also includes support for a variety of reasoners through the inference API.

In Jena API, ontology models are created through the Jena ModelFatcory as:

OntoModel m= ModelFactory.createOntologyModel();

This declaration creates an ontology model with the default settings. These defaults are:

- OWL-Full language
- In-memory storage
- RDFS inference.

There are several ontology languages including RDFS, DAML+OIL, OWL Full, OWL DL and OWL Lite. To create an ontology model for a particular language including the type of reasoner and storage model, OntoModelSpec is used. A number of common OntModelSpec constants are pre-declared and the common ones are listed in table 5.2.

Table 5 1: List of pre-declared OntModelSpec constants.

OntModelSpec	Language	Storage model	Reasoner
OWL_MEM	OWL full	In-memory	None
OWL_MEM_TRANS_INF	OWL full	In-memory	Transitive class-hierarchy inference
OWL_MEM_RULE_INF	OWL full	In-memory	Rule-based reasoner with OWL rules
OWL_MEM_MICRO_RULE_INF	OWL full	In-memory	Optimized rule-based reasoner with OWL rules
OWL_DL_MEM_RDFS_INF	OWL DL	In-memory	Rule reasoner with RDFS-level entailment-rules.
RDFS_MEM	RDFS	In-memory	None
OWL_LITE_MEM	OWL Lite	In-memory	None

5.2.4 Application

The fourth component of our design architecture is the application level. It is responsible for making queries on the integrated databases or to create integrated database.

6. Implementation

This section addresses the tools and development environment used in developing the system.

6.1 Implementation of the database sources

To test the applicability of our design, we considered database sources from the federal ministry of health. We created two databases one for Oromiya health office and the other for Addis Ababa health office using mysql database management system. Each database source contains tables listed in Table 6.1

Table 6. 1: List of tables in source databases

No.	Table	Description
1	HealthInstitution	Maintains data about distribution of health institutions owned by OGA, NGO and Private.
2	Drug Retailer	Maintains data about distribution of drug retailer outlets owned by OGA,NGO and Private
3	HealthProfessional	Maintains data about distrib uton of number of health professionals in Government, NGO, and Private institutions
4	FamilyPlanning	Maintains data about family planning undertaken by OGA,NGO and Private
5	Infrastructure	Maintains data about type of infrastructure undertaken

In the next subsections, the definitions of the schemas from the two databases are presented.

6.1.1`Addis Ababa Database schema definition

drugretaileroutlet CREATE TABLE drugretaileroutlet (*woreda char(15) NOT NULL,numberOfOgaPharmacy int(11) default NULL* *`numberOfOgaPharmacy int(11)defaultNULL,,`numberOfNgoPharmacy int(11)* *`numberOfPrivatePharmacy`int(11)defaultNULL, `* *numberOfRularDrugVendors int(11) default NULL,PRIMARY KEY (woreda))*
infrastructure CREATE TABLE `infrastructure (* *woreda` varchar(15) NOT NULL, `numberOfNewHospitals int(11) default NULL ` numberOfNewHealthCenters int(11) default NULL,, numberOfNewHealthPosts int(11) default NULL *numberOfExpandedHospitals int(11) default NULL ,* *numberOfExpandedHealthCenters int(11) default NULL,* *numberOfExpandedHealthPosts int(11) default NULL,,* *numberOfRehabilitatedHospitals int(11) default NULL,* *numberOfRehabilitatesHealthCenters int(11) default NULL,* *numberOfRehabilitatedHealthPosts int(11) default NULL,* *PRIMARY KEY (woreda))*
woreda CREATE TABLE woreda (*woredaName varchar(15) NOT NULL, numberOfOgaHospitals int(11) default*

```
numberOfNgoHospitals int(11) default NULL, numberOfPrivateHospitals int(11)
numberOfOgaSpecialClinics  int(11) default NULL,
 numberOfNgoSpecialClinics  int(11) default NULL,
 numberOfPrivateSpecialClinics  int(11) default NULL          ,
 numberOfOgaHigherClinics  int(11) default NULL,
 numberOfNgoHigherClinics  int(11) default NULL,
  numberOfPrivateHigherClinics  int(11) default NULL,
  numberOfOgaMediumClinics int(11) default NULL,
  numberOfNgoMediumClinics int(11) default NULL,
  numberOfPrivateMediumClinic  int(11) default NULL,
     PRIMARY KEY  (woredaName)
```

6.1.2 Oromiya Database Schema definition

CREATE TABLE drugretaileroutlet (*woredaName varchar(15) NOT NULL,* *numberOfNonGovOrgPharmacy int(11) default NULL,* *numberOfPrivatePharmacy` int(11) default NULL,* *numberOfRularDrugVendors int(11) default NULL, PRIMARY KEY ( woredaName`)*
CREATE TABLE `infrastructure (* *woreda varchar(15) NOT NULL, noOfNewHospitals int(11) default NULL noOfNewHealthCenters` int(11) default NULL,noOfNewHealthPosts int(11) default *noOfExpandedHospitals int(11) default NULL `noOfExpandedHealthCenters int(11) noOfExpandedHealthPosts int(11) default NULL,,noOfRehabilitatedHospitals int(11) default* *`noOfRehabilitatedHealthCenters int(11) default NULL, noOfRehabilitatedHealthPosts int(11) default NULL , PRIMARY KEY (woreda))*

```
CREATE TABLE woreda (

woredaName varchar(15) NOT NULL,numberOfOtherGovAuthorityHospitals int(11)

numberOfNonGovOrgHospitals  int(11) default NULL,

 numberOfPrivateHospitals  int(11) default NULL,

  numberOfOtherGovAuthoritySpecialClinics  int(11) default NULL,

numberOfNonGovSpecialclinics  int(11) default NULL,

 numberOfPrivateSpecialClinics int(11) default NULL,

 numberOfOtherHigherClinics  int(11) default NULL,

  numberOfNonGovOrgHigherClinics  int(11) default NULL,

  numberOfPrivateHigherClinics  int(11) default NULL,

  numberOfOtherOrgAuthorityMediumcliics   int(11) default NULL,

  numberOfNonGovOrgMediumClinics int(11) default NULL,

  numberOfPrivateMediumClinics int(11) default NULL,

    numberOfPrivateSpecialClinics int(11) default NULL,

   numberOfOtherHigherClinics  int(11) default NULL,

   numberOfNonGovOrgHigherClinics  int(11) default NULL,
```

6.2 Implementation of relational to RDF/OWL converter algorithm

As discussed in chapter 5, this algorithm automatically creates RDF/OWL file from relational database data by accepting database server name, database name and other attributes. Java programming language is used to implement the algorithm and figure 6.1 shows the interface for this algorithm. If appropriate data is filled to the interface figure 6.2 will be displayed and figure 6.3 shows the output created by the algorithm with the given sample data.

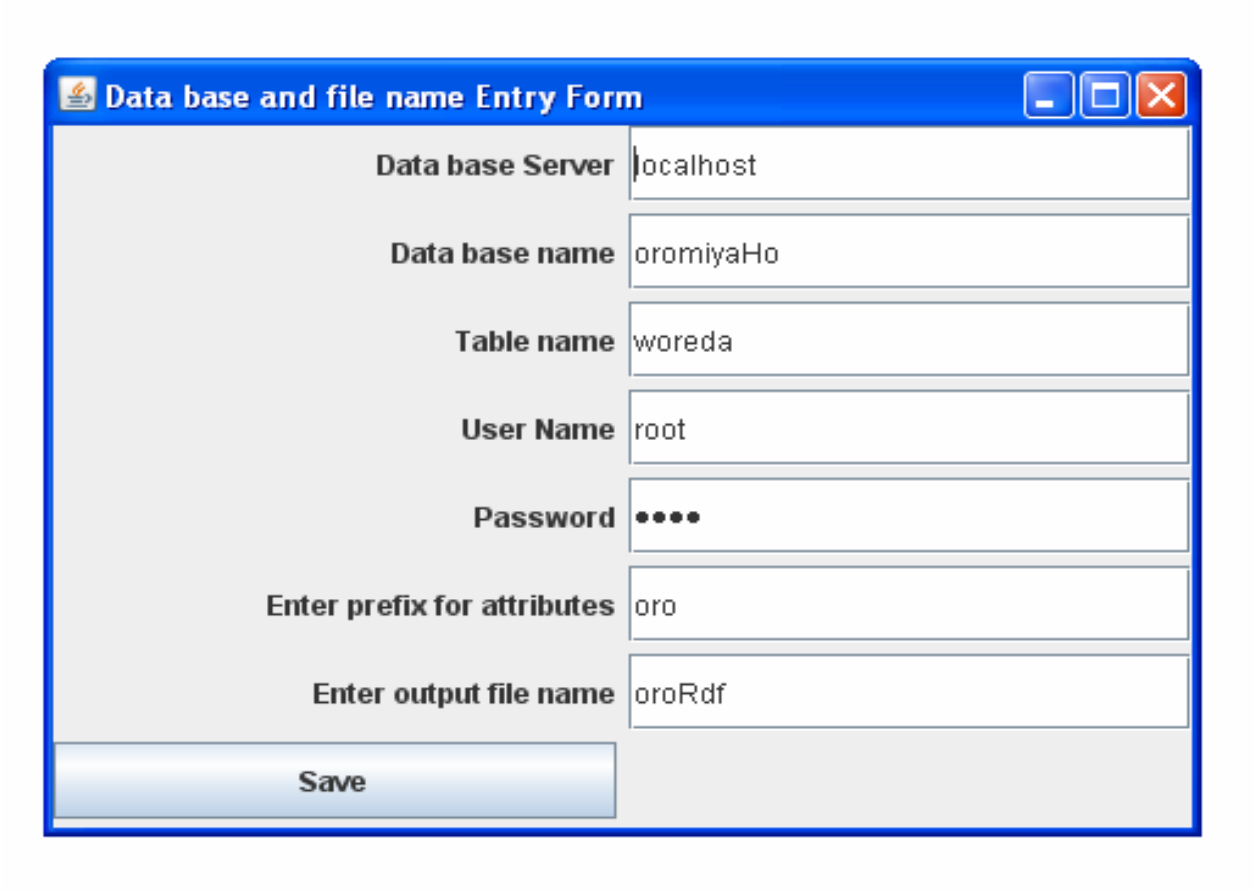

Figure 6.1: Interface to create RDF/OWL file from RDB

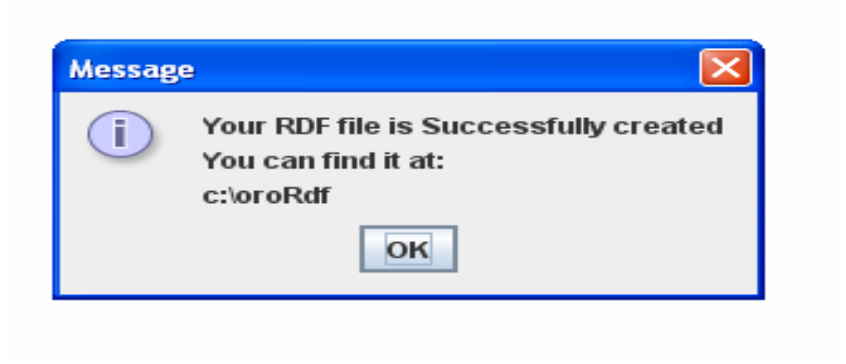

Figure 6 .2: Message showing the successful creation of RDF/OWL file.

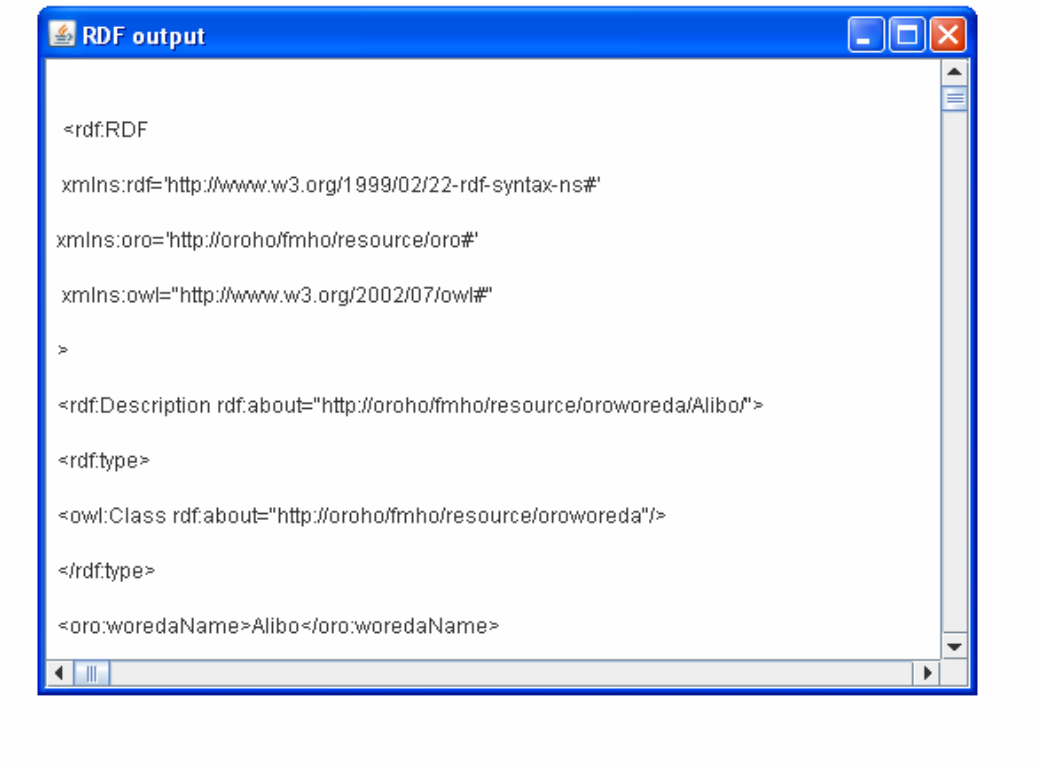

Figure 6. 3: Sample Output produced by the converter algorithm.

6.3 Implementation of Ontology

In order to provide an integrated view of two or more database sources, there must be a set of shared concepts, even if these concepts only cover a limited part of the information stored in the two resources. The concepts could even be referenced by different terms as long as they have the same semantics and a mapping is provided. Ontologies offer a uniform set of terms with which to refer to concepts throughout a domain. The concepts on ontologies are organized in line with how users conceive of their domain rather than the data structures created to conveniently store the information. The ontology is used as an interchange format. .

For a given application, ontologies that are already built by domain experts can be imported, refined, and extended or built from scratch. For this project, we developed ontology that meets our need using protégé 3.2 as our development tool. Figure 6.3 shows list of the classes we defined in the ontology. Classes are concepts in the domain of discourse and they are described by attributes or properties or slots. Classes have instances and properties have restrictions. Building ontology then means:

- defining classes in the ontology,
- arranging the classes in a taxonomic (subclass–superclass) hierarchy,
- defining slots and describing allowed values for these slots,
- filling in the values for slots for instances
- defining restriction on slots.

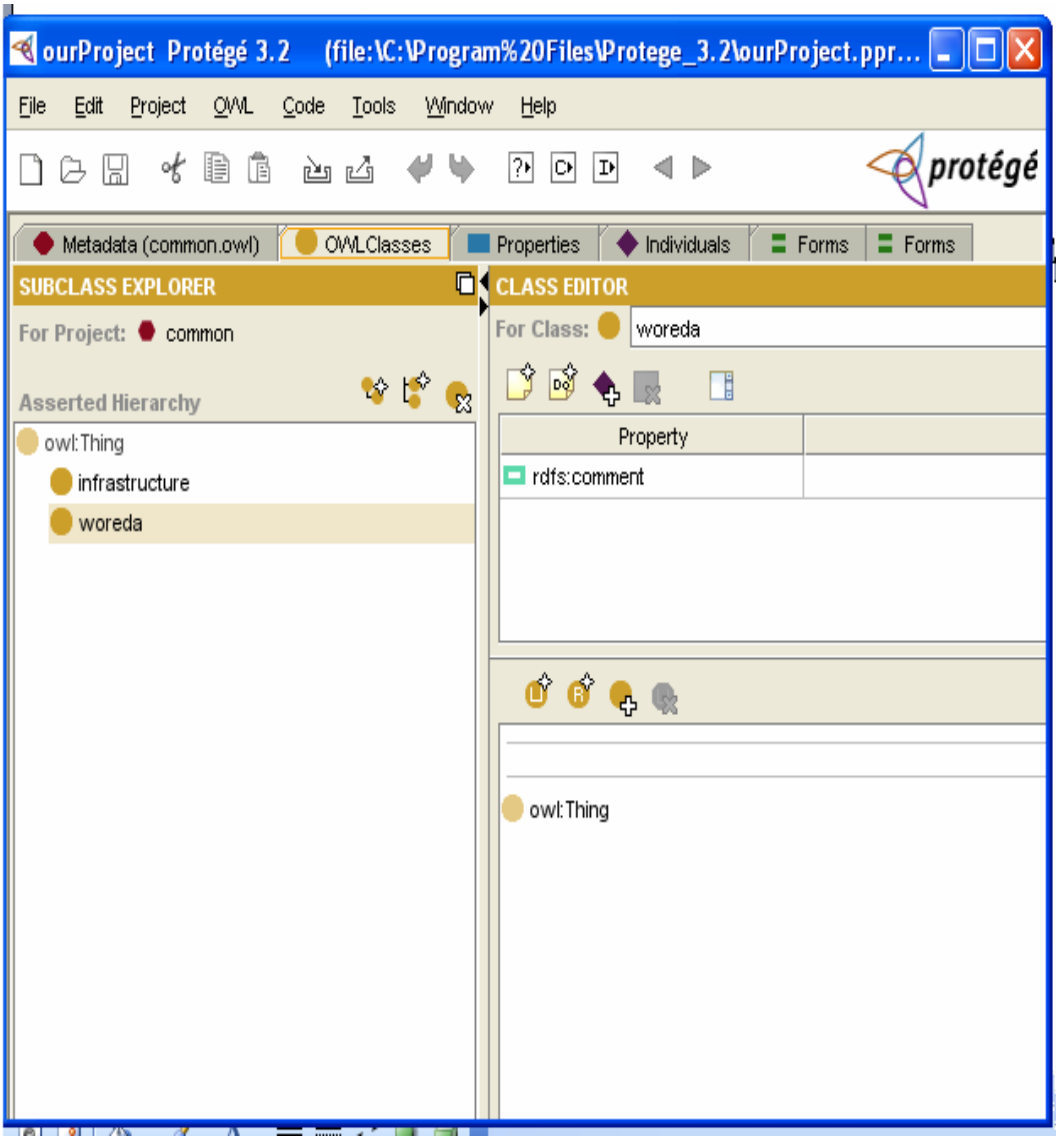

Figure 6 .4: List of classes in the ontology.

By following the above steps, we built our ontology and list of properties defined for the chosen classes are shown in figure 6.4.

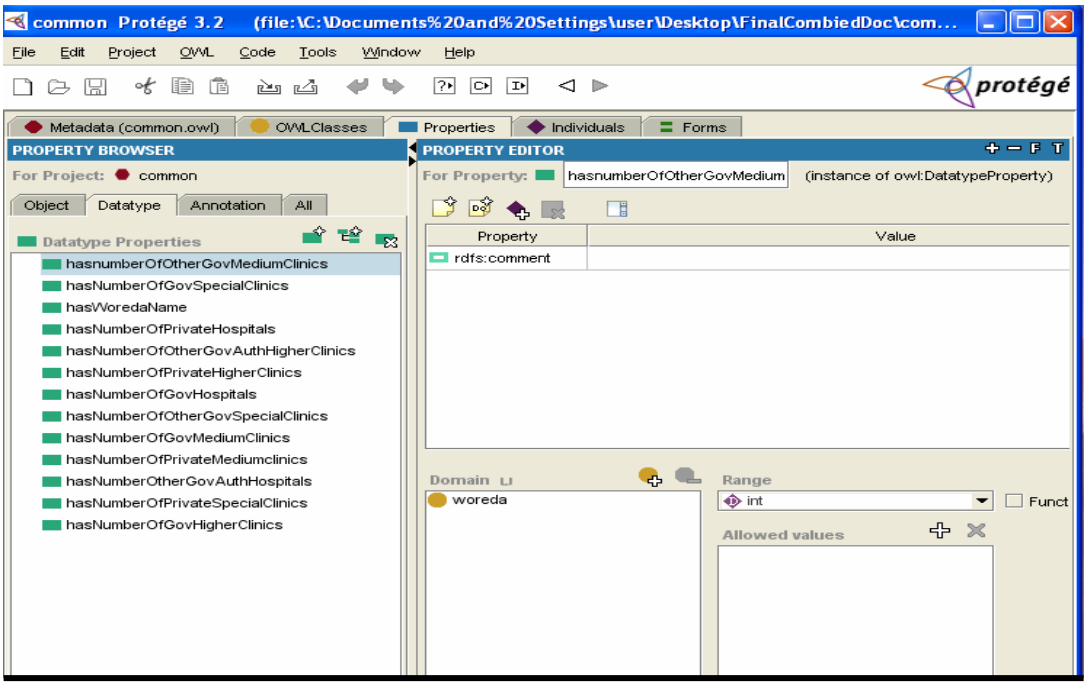

Figure 6. 5: list of properties

After the ontology is developed, the next step is to map concepts on the combined RDF/OWL files to concepts on the ontology. During this mapping, table names from the databases are mapped to corresponding classes and table attributes are mapped to properties or slots on the ontology. If two attributes from the two databases have the same semantic (represent the same concept) they are mapped to the same property on the ontology. Figure 6.5 shows the mapping process.

Forexample,hasNumberOfNgoSpecialClinics is one property on the ontology (shown to left of the screenshot) to indicate that an instance of woreda class has number of non governmental special clinics. Table fields *numberOfNgoSpecialClinics* from table *addisWoreda* of *addisAbabaHo* database and *numberOfNonGovSpecialclinics* from table *oroWoreda* of *oromiyaHo* database represent the same concept as the property

hasNumberOfNgoSpecialClinics in the ontology. Therefore, these two fields are mapped to the property *hasNumberOfNgoSpecialClinics* using equivalent construct.

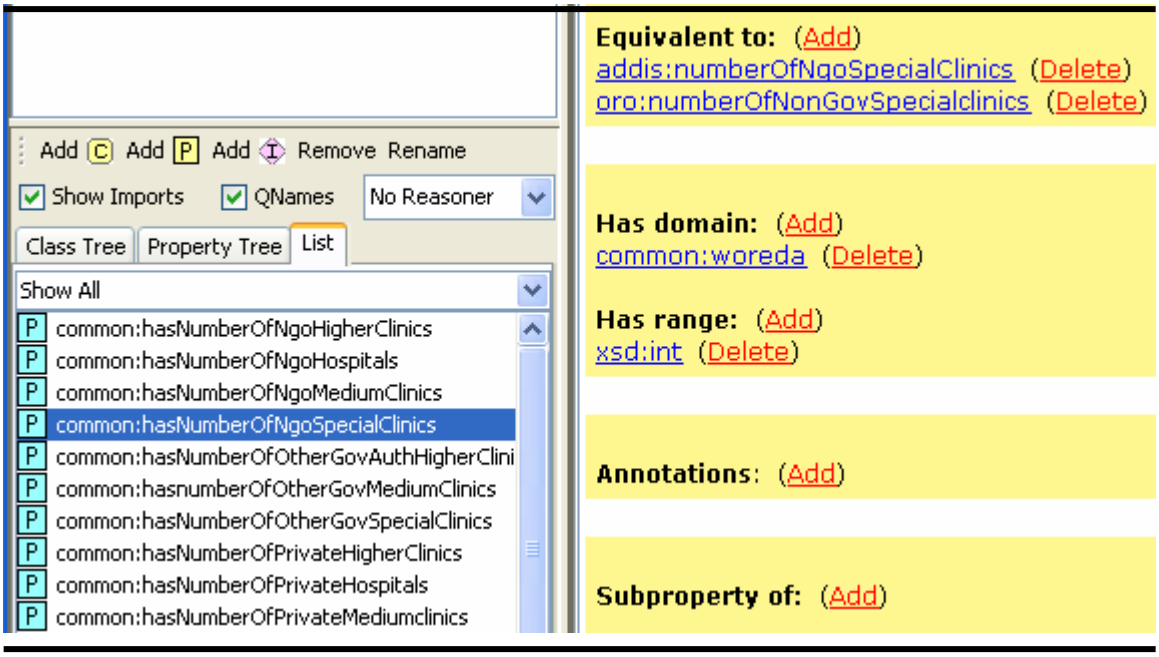

Figure 6.6: Mapping equivalent fields to the same property.

6.4 Application

After semantic conflicts are resolved through the mapping process, the next step is to develop an application on the combined database sources. One possible application is to write queries on the combined database sources. In this sub section, we will describe the tools we used to write queries and display their results. Sample queries we wrote on the combined database sources are also described.

To retrieve and display results of all the queries, we used Jena 2.5.5, which is a Java framework for building semantic web applications. Jena provides programming environment for SPARQL query language.

SPARQL is a syntactically SQL like language for querying RDF graphs via pattern matching. The languages features include basic conjunctive patterns, value filters, optional patterns, and pattern disjunction. It is the product of W3C's Data Access Working Group.

SPARQL only queries the information held in the models; there is no inference in the query language itself. That is, SPAQL does not do anything other than take the description of what the application wants, in the form of a query, and returns that information, in the form of a set of bindings or an RDF graph.

SPARQL queries RDF graphs. An RDF graph is a set of triples (graph "models" and triple "statements").

For inference, we used the OWL reasoning provided by Jena.The following declaration creates an instance of the OWL reasoner specialized to the application we wanted to write.

Model schema = *FileManager.get().loadModel("file:data/ourOntology.owl");*

Model data= *FileManager.get().loadModel("file:data/ourRdfFile.rdf");*

Reasoner reasoner=*ReasonerRegistry.getOWLReasoner();*

reasoner = *reasoner.bindSchema(schema);*

InfoModel infomodel=*ModelFactory.createInfoModel(reasoner,data);*

Query query = *QueryFactory.create(queryString, Syntax.syntaxARQ);* Where

Model,Reasoner ,InfoModel and Query are classes provided by Jena API.

The following queries show sample queries that we wrote on the combined database.

Query 1: List all woreda names from the two databases.

Solution:

```
SELECT DISTINCT   ?y
            WHERE {
              ?x   http://www.fmo.gov/common#hasWoredaName ?y.
                 }
```

In this query, *hasWoredaName* is a property on the ontology.But in the two databases, *addis:woredaName* and *oro:woredaName* are fields mapped to this property. Because of the mapping, the above single query displays results from the two databases and the result of this query is shown in figure 6.7.

Figure 6.7: Results of query 1

Query 2.List all woredas with their corresponding number of private higher clinics from the two databases.

Solution:

```
SELECT   ?y  ?z
   WHERE {
  ?x <http://www.fmo.gov/common#hasWoredaName> ?y.
 ?x <http://www.fmo.gov/common#hasNumberOfPrivateHigherClinics> ?z.
 }
```

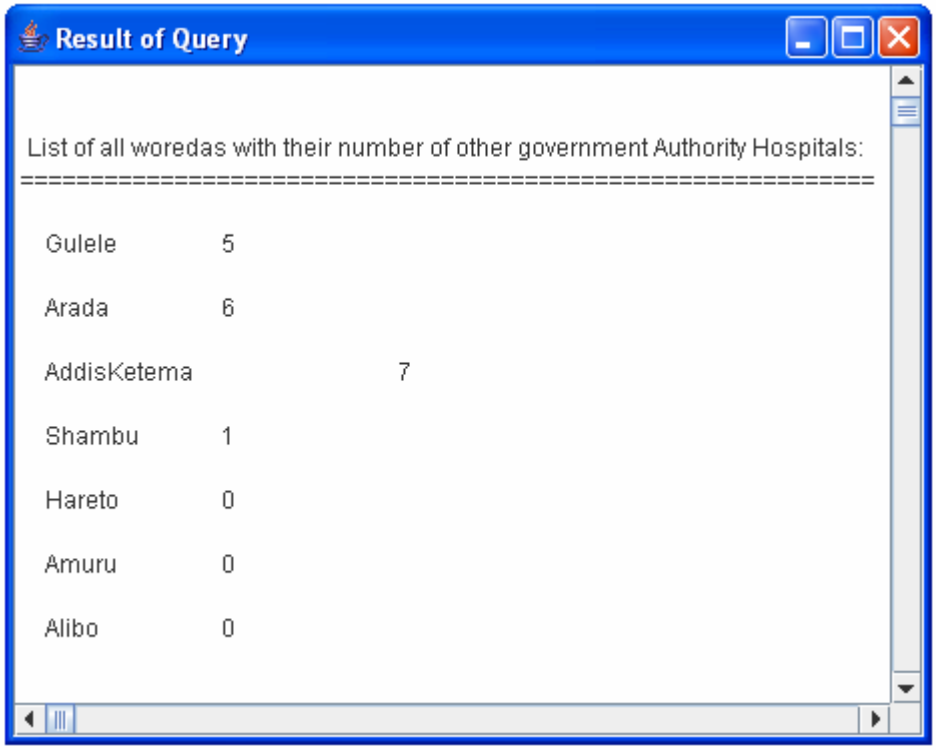

Figure 6 .8: Results of query 2

7. Conclusion and Future Work

7.1 Conclusion

Database integration is one of the important research issues in information system. In this project, we have treated some of the several database integration challenges.

We have proposed semantic database integration architecture with four components: Data sources, Database to RDF Gateway, Mapping and Integration Engine, and Application.

The Database sources component contains those databases that are going to be integrated. The Database to RDF Gateway converts database schemas to RDF/OWL .The Mapping and Integration Engine reasons and integrates multiple schemas to RDF schemas.

To test the feasibility of our architecture, we have developed a prototype using heterogeneous database sources from the FMOH. We have created and developed two sample database sources and a common ontology that defines the semantic and relationships of the schemas of the source databases.

Algorithm that maps source database schemas to RDF/OWL is implemented. We have used the JENA RDF/OWL reasoning tools to reason and integrate multiple RDF schemas created for each database source to a single RDF schema. Finally, by setting sample queries we have tested the system for retrieval of data from all the sources. The result output from the databases proves to be correct and it meets our integration requirements.

In conclusion, the result is encouraging and satisfactory. It showed that automatic and semantic integration of large scale heterogeneous databases for federated environment can be handled using our approach.

7.2 Future works

As we have discussed in the related works part of this paper, there are several forms of heterogeneity among databases sources. We believe that our design architecture can resolve any of the heterogeneity forms as long as the common ontology and additional custom rules are well defined and provided to the mapping and Integration engine. But, since in our prototype development, we have limited our scope and defined the ontology in such away that it can only resolve naming conflicts, the other heterogeneity forms could not have been handled. Therefore, further extensions are necessary to modify the ontology and define additional custom rules as necessary to handle the other forms of heterogeneity.

8. References

[1] Ph. Thiran and J-L. Hainaut . Interoperability of Legacy Databases .A Combined Top-Down and Bottom-Up Approach.PhD thesis.

[2] Hainaut,J-L. Hainaut: “Specification preservation in schema transformations - Application to semantics and statistics”, *Data & Knowledge Engineering*, Elsevier Science Publish, 16(1), 1996.

[3] A.P. Sheth and J.A. Larson .“Federated Database Systems for Managing Distributed, Heterogeneous and Autonomous Databases”, *ACM Computing Surveys*, 22(3):183-236, September 1990.

[4] Hammer, J., and D. McLeod. An Approach to Resolving Semantic Heterogeneity in a Federation of Autonomous, Heterogeneous Database Systems. International Journal of Intelligent & Cooperative Information Systems 2:51-83,1993.

[5] Kim, W.Modern database systems : the object model, interoperability, and beyond. ACM Press ; Addison-Wesley Pub. Co., New York, N.Y., Reading, Mass,1995.

[6] Wang X., Gorlitsky R., and Almeida, J. S. From XML to RDF: “how Semantic Web technologies will change the design of standards.” *Nat Biotechnol* 23(9): 1099-103,2005.

[7] E.-P. Lim, etal. Entity identification in database integration. In *International Conference on Data Engineering*, pages 294–301, Los Alamitos, Ca., USA, April 1993. IEEE Computer Society Press.

[8] Davidson, etal. Challenges in integrating biological data sources. J Comput Biol 2:557-572,1995.

[9] Karp, P. D.A Strategy for Database Interoperation. J Comput Biol **2**:573-586,1995.

[10]T.Hernandez and S. Kambhampati,Integration of Biological Sources: CurrentSystems and Challenges Ahead,SIGMOD Record, 33 (2004)

[11] Buneman P,etal. A Data Transformation System for Biological Data Sources, in Proc. 21st Int. Conf VLDB. 158-169, 1995

[12]Cheng Hian Goh. Representing and Reasoning about Semantic Conflicts in Heterogeneous Information Sources. Phd, MIT, 1997.

[13] Uschold and Gr¨uninger. Ontologies: Principles, methods and applications.Knowledge Engineering Review, 11(2):93–155,1996.

[14] H. Wache,etal.Ontology-Based Integration of Information—A Survey of Existing Approaches

[15] DARPA Agent Markup Language Homepage, http://www.daml.org/about.html , Accessed on May 2008

[16] Horrocks, et al., OIL in a Nutshell,Proc. ECAI 2000 Workshop on Application of Ontologies and PSMs, Berlin, Germany, 2000, pp. 4.1-4.12.

[17] Boris Motik, etal. OWL 1.1 Web Ontology Language Structural Specification and Functional-Style Syntax.http://www.w3.org/Submission/owl11-owl_specification,Accessed on May 2008.

[18] Schema Integration Methodologies for Multidatabases and the Relational Integration Model - Candidacy document

[19] Bemers-Lee T., Hendler J., and Lassila O. The Semantic Web. Scientific American. 284(5): 34-43, 2001.

[20] Donis-Keller H,et al. A Genetic Linkage Map of the Human Genome. *Cell.* 51: 319-337, 1987.

[21] Baader F., Calvanese D., McGuinness D., Nardi D., and Patel-Schneider P. The Description Logic Handbook.Cambridge University Press, 2002.

[22] J. Barrasa, Ó. Corcho, A. Gómez-Pérez: R2O, an Extensible and Semantically Based Data-baseto-ontology Mapping Language,Second Workshop on Semantic Web and Databases (SWDB2004). Toronto, Canada,August 2004.

[23] Bizer C. D2R MAP – A DB to RDF Mapping Language. 12th International World Wide Web Conference, Budapest,May 2003

[24] Barrasa J, Corcho O, Gómez-Pérez A.FundFinder – A case study of Database-toontology mapping,Semantic Integration Workshop, ISWC 2003,Sanibel Island, Florida,Sept 2003

[25] Návrat, P., Bieliková, M., Rozinajová, V.:Methods and Tools for Acquiring and Presenting Information and Knowledge in the`Web.CompSysTech 2005, B. Rachev, A. Smrikarov (Eds.), Varna, Bulgaria, June 2005. pp. IIIB.7.1-IIIB.7.6.

[26] EPrud'hommeaux and A. Seaborne.SPARQL Query Language for RDF. http://www.w3.org/TR/2006/CR-rdf-sparqlquery-20060406/, 2006.