

**FEDERATED DATABASE FRAMEWORK FOR DISEASE
OUTBREAK INFORMATION AND NOTIFICATION
SYSTEMS: A WEB SERVICE APPROACH**

BY

MUSTAFA GHALEB

A Thesis Presented to the
DEANSHIP OF GRADUATE STUDIES

KING FAHD UNIVERSITY OF PETROLEUM & MINERALS

DHAHRAN, SAUDI ARABIA

In Partial Fulfillment of the
Requirements for the Degree of

MASTER OF SCIENCE

In

COMPUTER SCIENCE

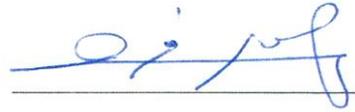
DECEMBER, 2014

KING FAHD UNIVERSITY OF PETROLEUM & MINERALS

DHAHRAN- 31261, SAUDI ARABIA

DEANSHIP OF GRADUATE STUDIES

This thesis, written by **Mustafa Ghaleb** under the direction his thesis advisor and approved by his thesis committee, has been presented and accepted by the Dean of Graduate Studies, in partial fulfillment of the requirements for the degree of **MASTER OF SCIENCE IN COMPUTER SCIENCE**.



Dr. FARAG AHMED AZZEDIN
(Advisor)



Dr. ADEL AHMED
Department Chairman



Dr. SAJJAD MAHMOOD
(Member)



Dr. Salam A. Zummo
Dean of Graduate Studies



Dr. HASSINE JAMELEDDINE
(Member)

28/1/15

Date

© Mustafa Ghaleb

2014

Dedication

I would like to dedicate this work to my parents, wife, daughter, son, brothers and sisters.

This work would not be possible without their support. I would also like to thank all of my previous teachers, as their hard work has laid the foundation for this work.

ACKNOWLEDGMENTS

First I would like to thank almighty Allah for giving me the ability and the strength to work on and complete this thesis. I would like to give my sincere appreciation to my advisor Dr. Farag Azzedin for to all the hard work he has put in this work, and for the guidance and encouragement he provided throughout this research. Dr. Farag has been a mentor, teacher and a father. I feel very lucky to have had Dr. Farag as my thesis supervisor. I would also like to thank Dr. Farag for helping me in my future endeavors.

I would also like to thank the committee members, Dr. Sajjad Mahmood and Dr. Hassine Jameleddine, for dedicating time out of their busy schedule for this work and providing their feedback.

I also would like to express thanks to Mr. Jaweed Yazdani for his valuable advices and guidance. I also acknowledge KFUPM and information and computer science (ICS) department as well as all faculties who taught me.

Lastly, I would like to thank Muaadh Abdulghani, and Taher Ghaleb for their encouragement and moral support.

TABLE OF CONTENTS

ACKNOWLEDGMENTS	V
TABLE OF CONTENTS.....	VI
LIST OF TABLES.....	VIII
LIST OF FIGURES.....	IX
LIST OF ABBREVIATIONS.....	X
ABSTRACT.....	XI
ملخص الرسالة.....	XIII
CHAPTER 1 INTRODUCTION.....	1
1.1 Overview	1
1.2 Problem Statement.....	6
1.3 Thesis contribution	8
1.4 Thesis Outline	9
CHAPTER 2 LITERATURE REVIEW	10
2.1 Disease Outbreak Notification Systems (DONS)	10
2.2 Database Federation.....	14
2.2.1 Characterization of federated databases	15
2.2.2 Strengths and Weaknesses	16
2.2.3 Federated Databases in Healthcare Systems.....	17
2.3 Web Services.....	19
2.4 Web Services in Healthcare Systems	27

CHAPTER 3 THE PROPOSED FRAMEWORK.....	29
3.1 Methodology of implementing web services integration	30
3.2 Framework Layers.....	32
3.3 Framework Workflow	35
3.4 Proposed Approach Features	38
CHAPTER 4 IMPLEMENTATION OF THE PROPOSED FRAMEWORK.....	40
4.1 Overview	40
4.2 Development Environment Setup	42
4.2.1 Prototype Implementation Architecture	43
4.2.2 Databases	47
4.2.3 Data Services Server	54
4.2.4 Experiments.....	55
4.2.5 An Example of data service	65
4.2.6 An Example of a WSDL file for one WS.....	69
CHAPTER 5 CONCLUSION AND FUTURE WORK	79
REFERENCES.....	81
VITAE.....	87

LIST OF TABLES

Table 1 : ‘‘SOAP-based’’ and ‘‘RESTful’’ Web services.....	25
Table 2: approaches to implement integration of web services	30

LIST OF FIGURES

Figure 1 : Accessing each healthcare database for collecting data.....	7
Figure 2: Basic structure of federating system	14
Figure 3 : The three roles in Web service application	20
Figure 4 : Layers of DONSFed.....	33
Figure 5 : Framework Workflow	36
Figure 6 : Query partitioning to send each sub-query to the specific component DB.....	37
Figure 7 : DONSFed Prototype System Architecture.....	44
Figure 8 : Deployment diagram of DONSFed.....	46
Figure 9 : Complete KSA DONS database schema on the Oracle platform	48
Figure 10 : Complete CASE database schema on the MySQL platform.....	50
Figure 11 : MySQL database migration to SQL Server using SSMA tool.....	52
Figure 12 : Complete DONS database schema on the SQL Server platform	53
Figure 13 : DONSFedOracle data service	56
Figure 14 : DONSFedMySql data service	57
Figure 15 : DONSFedSQLServer data service	58
Figure 16 : DONSFed home page.....	59
Figure 17 : Federated service for case identification by statistics date.....	60
Figure 18 : Federated service for disease identification by disease name	62
Figure 19 : Federated service for getting all Cases which their infection type is Chronic	62
Figure 20 : Federated service for case identification by the country of infection	63
Figure 21 : Federated service for case identification by its diagnosis	64

LIST OF ABBREVIATIONS

DONS	:	Disease Outbreak Notification System
DONSFed	:	Federated System for Disease Outbreak Notification Systems
XML	:	Extensible Markup Language
WSDL	:	Web Services Description Language
SOAP	:	Simple Object Access Protocol
REST	:	Representational State Transfer
UDDI	:	Universal Description, Discovery and Integration
HTTP	:	Hypertext Transfer Protocol
WS	:	Web Service

ABSTRACT

Full Name : Mustafa Mohammed Saeed Ghaleb
Thesis Title : Federated Database Framework For Disease Outbreak Information
And Notification Systems: A Web Service Approach
Major Field : Computer Science
Date of Degree : December 2014

Nowadays, the volume of healthcare data, which is increasing exponentially, is stored in geographically distributed databases. Healthcare practitioner, epidemiologists and researchers who want to obtain critical information from these databases must access multiple databases for collecting and analyzing the information to make critical decisions related to their work. However, this can be a time-consuming and error-prone process. Providing a uniform and logical unit to obtain data is not an easy task and needs clean data integration process.

Among the data integration approaches, Federated Databases and Web Services are among latest approaches that minimize the interference of current operations, handle the heterogeneities, maintain the local autonomy of component systems and are scalable.

In this thesis, we proposed and designed a federated databases framework for outbreak information and notification systems. This framework was implemented by using web services as an integration platform. We also provided a proof of concept prototype implementation of the proposed approach.

For providing proof-of-concept implementation to the proposed framework, we used three different autonomous and distributed databases. The first one was obtained from the

KSA DONS system which is an oracle cloud-based database. The second one was acquired from the CASE system which is the open source MySQL database, while the third database is based on SQL Server. These databases which are located at different locations, have different schemas, and different semantics, and are therefore suitable for testing our federation framework. Finally, the experimental results are presented and analyzed in this thesis.

ملخص الرسالة

الاسم الكامل: مصطفى محمد سعيد غالب

عنوان الرسالة: إطار قاعدة بيانات اتحادية لأنظمة التبليغ والمعلومات للأمراض المتفشية باستخدام تقنية خدمات الويب

التخصص: علوم حاسوب

تاريخ الدرجة العلمية: ديسمبر 2014

في الوقت الراهن، ونتيجة للإزدياد المطرد، يتم تخزين بيانات الرعاية الصحية في قواعد البيانات الموزعة جغرافياً، حيث يجب على ممارسي الرعاية الصحية والباحثين- الراغبين في الحصول على المعلومات الهامة من قواعد البيانات تلك -الوصول إلى قواعد بيانات متعددة حتى يتمكنوا من جمع وتحليل المعلومات ومن ثم اتخاذ قرارات حاسمة تتعلق بعملهم.

ومع ذلك، يمكن أن تكون هذه العملية مضيعة للوقت وعرضة للأخطاء، وعليه فإن توفير وحدة مركزية ومنطقية للحصول على تلك البيانات ليست مهمة سهلة وتحتاج إلى معالجة البيانات بشكل سليم وتكاملي.

ومن نظريات تكامل البيانات، فإن طريقة قواعد البيانات الاتحادية وخدمات الويب تعد من بين أحدث الأساليب التي تقلل من التداخل في العمليات المستخدمة حالياً، وتتعامل مع التباين، والحفاظ على التحكم الذاتي المحلي للعناصر المكونة للنظام الاتحادي وتكون قابلة للتطوير . في هذه الرسالة، اقترحنا تصميم إطار لقاعدة بيانات اتحادية لأنظمة التبليغ عن الأمراض المتفشية. وقد نفذ هذا الإطار باستخدام خدمات الويب باعتباره بيئة تكاملية. كما أثبتنا بالدليل كيفية تنفيذ مفهوم النموذج الأولي للإطار المقترح.

ولإثبات صحة مفهوم تنفيذ الإطار المقترح، استخدمنا ثلاث قواعد بيانات مختلفة ومستقلة وموزعة. الأولى: تم الحصول عليها من نظام DONS KSA ، وهي قاعدة بيانات تستند إلى قاعدة بيانات أوراق القائمة على الحوسبة السحابية. والثانية: من نظام CASE التي هي قاعدة بيانات مفتوحة المصدر تستند على قاعدة بيانات MySQL، بينما تستند قاعدة البيانات الثالثة على SQL Server. وتتموقع قواعد البيانات هذه في أماكن مختلفة ولها مخططات ودلالات مختلفة، وبالتالي فهي مناسبة لاختبار الإطار المقترح والتحقق من صحته. وفي الأخير، قمنا بعرض نتائج التجارب التي قمنا بها و تحليلها في هذه الرسالة .

CHAPTER 1

INTRODUCTION

1.1 Overview

In the last several years, the quantity of healthcare data has increased exponentially. Progressive growth in data related to healthcare brings a challenge to manage it [1]. One of the major areas of the health care field is outbreak information and notification systems which involve extensive data management.

Outbreaks such as SARS [2][3], Mad Cow diseases (BSE), different strains of "bird flu" or Influenza (i.e., H1N1 and H5N1) [4], along with the threat of bio-terrorism (e.g. anthrax) [4][5] are the most intriguing and complex phenomena that confront scientists in the field of microbiology, virology and epidemiology [7]–[9]. The ability of these viruses to mutate and evolve is one of the big mysteries that puzzle health officials, who are trying to find out the root cause of worldwide pandemics since the late 1880s. Pandemics occur when small changes in the virus over a long period of time eventually “shift” the virus into a whole new subtype, leaving the human population with little or no time to develop immunity.

Therefore, a tremendous and an unforeseen threat could mark the start of a global outbreak given the above-mentioned properties, namely (a) the ability of the virus to mutate into a whole new subtype, (b) the acute time shortage for the human population to

develop immunity, (c) the limitation in terms of the effect of immunization, and (d) the viruses' ability to change to a form that is highly infectious for humans and spread easily from one person to another. Furthermore, as outlined by WHO and the World Organization for Animal Health (OIE) [10][11], the international standards, guidelines and recommendations in an event of an outbreak state that "Member Countries are obliged to notify within 24 hours epidemiological information with regards to occurrence/reoccurrence of listed notifiable diseases, the occurrence of a new strain of a listed disease, a significant change in the epidemiology of a listed disease, or the detection of an emerging disease" [12]. It is apparent that the increasing threat of disease outbreak highlights the need to provide timely and accurate information to public health professionals across many geographical, jurisdictional and organizational boundaries.

Basically exchanging health data today may include accessing and analyzing a wide variety of formats, data models, data language, and protocols that go beyond just row data. To achieve accessing and managing of variety of computerized processes, it is necessary for an organization to have database systems that can interchange and operate over a distributed and heterogeneous network. Consequently, aggregating and querying data from heterogeneous systems has become a hot research area among information researchers [13].

In this thesis, we focused on the federated method. Under this approach local database systems can continue their local transactions and operations without altering the supported features and services of local databases; but at the same time contribute in the federation due to the fact that this method is more reliable and stable in our case.

This federated method offers transparent query access and aggregates the results from multiple heterogeneous sources, including RDBMS, Triplestore, XML and NoSQL databases without the need to transform or rewrite the local data. Most studies in heterogeneous distributed systems only focus on RDBMS or a single language [13].

One key issue is the lack of integration between various information systems that need to be accessed for decision support by multiple stakeholders, both frontline health professionals and policy makers alike. Providing timely and accurate information requires both access and integration of databases, along with the ability to deliver medical disease data including, text, graphics and images, to a variety of terminal devices in the field. Medical disease data could be available in many distributed remote nodes. These remote nodes are health units, medical organizations or hospitals for which integration is needed. This requires access to distributed remote capabilities that handle high-speed data transfers, servers capable of handling such requests, terminals with capability to display the rich information, and resource sharing methods to make use of the distributed capacity. Unfortunately, the existing infrastructure makes it an implausible task and major replacement would be too costly.

The tremendous variety of healthcare data significantly improves a researchers' ability, in the healthcare field, to investigate the interactions between the constituents of the diseases, remedies and the medical disease identification systems. So, researchers need access to many geographically, different, distributed databases. Thus, a unified technique is needed for combining the heterogeneous data sources and databases [14][15].

Integration of data is a crucial technique for merging data from various disparate data sources. The data are stored and located at various locations in heterogeneous database packages which would need to be jointed to satisfy certain rational reasons such as collaboration between various institutions and organizations, co-operation between departments of the same organization and merging of two companies. Integration of data is not easy since multiple types of database management systems (DBMS) show substantial heterogeneity in their schema, query processing, data models, semantics, etc. Heterogeneity on database integration often includes joining data that exist among different data sources and showing data to end-users as a unified view of data [16].

Integration of data from multiple sources has become one of the major challenges and opportunities in enterprise computing today. With the acceleration in electronic production, large enterprises have valuable information stored in a number of systems. Furthermore, the World Wide Web can be seen as an endless source of data as millions of web pages emerge every day and the ultimate goal is to integrate such data to serve scientific decision making and operation. Based on this perspective, healthcare data integration is essential to query and retrieve valuable information from many local or remote databases which is a time-consuming process otherwise [17]–[20].

There are many methods to solve the data integration problems in information and health systems. The first method is manual encoding, where the programmers write codes for two programs needing integration and sharing of data. This manual encoding is a solution for single point and although it appears as a cheap solution, was verified that this solution consumed more time and proved expensive. The drawback of this method is that data integration cannot work seamlessly among various application systems [21]. However,

such methods of data integration have been utilized to solve the data integration problems.

Additionally, we summarize five data integration methods discussed in [22] and [1] as follows:

- **Link Integration** approach where users search starting from the first resource using hyperlinks to get relevant and comprehensive information. This approach has many problems such as instability of hyperlinks that are maintained by different hosts, ambiguities, and the vulnerability of naming conflicts [1].
- **Query-Based Integration** means that the users can query and retrieve data from multiple sources through a single query. The main problems in this method are the complexity of queries (e.g. not using the SQL) and the data location and integration are not transparent to users [23].
- **Data Warehouse Integration** means the systems can query and retrieve data from multiple resources to a unified, central repository. It provides many advantages such as improved performance, increased data consistency, access to multiple centralized and integrated databases. The major issues with this method include how to keep the updated data in a central repository, to have data scalability, and maintain privacy [24][25].
- **Federated Database Integration** is another approach that provides the users with uniform and central access to query and retrieve data [23]. It is more scalable and flexible than data warehouses because there is no need

for centralized repository. Hence, data replication is not required and it reduces problems such as data privacy and scalability.

- **Web Service Integration** has the possibility to become a broadly applicable integration method. It provides the extensibility and flexibility necessary for integration. Many healthcare databases today provide web services [23][26][27].

Among the data integration approaches, Federated Databases and Web Services are the prominent methods because they minimize the interference of current operations, handle the heterogeneities, maintain the local autonomy of component systems and are scalable. From the above-mentioned techniques and their problems of autonomy and heterogeneity, we propose to build a federated databases framework for outbreak information and notification system using web services.

1.2 Problem Statement

Nowadays, the volume of healthcare data, which is increasing exponentially, is stored in geographically distributed databases. Researchers who want to obtain critical information from these databases must access multiple databases for collecting and analyzing the information to make decisions about their work as shown in Figure 1 . However, this can be a time-consuming and error-prone process. Providing researchers with a uniform and logical unit to obtain data is not an easy task and needs clean data integration process. For the integration process, a federated database approach addresses these problems and it is suitable to many fields.

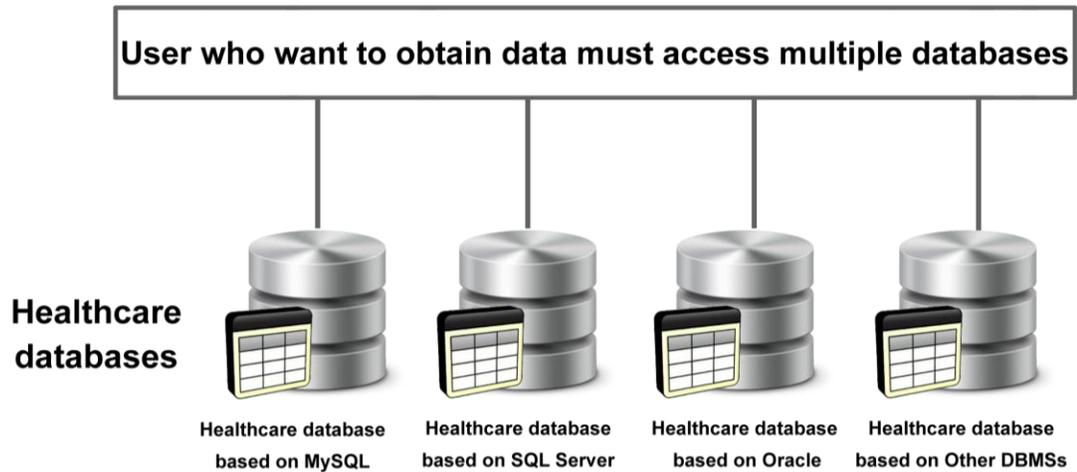


Figure 1 : Accessing each healthcare database for collecting data

The idea of federated databases is to integrate Database Management Systems (DBMSs) and databases without changing or discarding the component systems. This integration process should be implemented with the component databases without altering their management and functionality because they are in active use.

In traditional federated database approach, great time and effort are required to add new data source to the federation system or modify the current services that offered by a federated system. So, basic federated system introduces a latency that cannot be accepted in real time DONS systems that require quick response and early notification process. It also requires local to global schema translation to eliminate the data model heterogeneity among various component database systems. In order to maintain local autonomy of constituent databases, the traditional federated database approach requires deep knowledge of local schema for each component system. This may not be feasible in some systems such as DONS where some confidential information cannot be exposed to non-authorized personals.

These problems have motivated us to research the application of techniques to integrate DONS. To this end, a federated database system for outbreak information and notification systems using web services can provide the means which support the robustness and the autonomy of the developed system and allow adding new systems to the federation easily. As a consequence of this research, we have developed a framework that enables sharing and integration of heterogeneous data for the collaborative decision support DONS that requires quick response times.

Our federated database approach using Web services must support sharing and integration of heterogeneous data between various DONS, described in a layer-structured framework. The users access the portal of DONS federation as though all the functions and data come from one site. The portal accepts queries from the users and communicates with the component systems through the federation layer. Developing such a federated framework has some advantages in data sharing, data integration, performance and system expansion. It also removes the overhead of accessing distributed, heterogeneous data sources from each component system and enables a unified querying over federated data sources from a single access point.

1.3 Thesis contribution

- A detailed literature review of the data integration approaches: Here, the contribution of the thesis provided a detailed document surveying existing data integration methods and justification for the selection of web services as the data integration method used in this work.

- A framework of data integration approach suitable for disease outbreak information and notification system: Here, the contribution of the thesis includes design of a federated database framework. This framework is implemented using web services as an integration method.
- Proof-of-concept prototype implementation of the proposed approach: Here, the contribution of the thesis is a prototype provided as a proof of concept and to validate our framework.

1.4 Thesis Outline

The rest of the work is organized as follows. Chapter 2 presents a literature review of disease outbreak notification systems, federated databases and web services. It provides some examples using such an approach in healthcare systems. Chapter 3 contains the proposed framework for creating federated database for disease outbreak information and notification systems using web services. Chapter 4 highlights the environment setup as well as it provides a proof of concept implementation of DONSFed (our implementation of federated database). Some federated services are also added and implemented in this work. In addition, we added an example of one WSDL file and one complete data service. Finally, Chapter 5 concludes the thesis and presents some directions in which future work can be done.

CHAPTER 2

LITERATURE REVIEW

In this chapter, we reviewed Disease Outbreak Notification Systems (DONS) that cover various geographies and we mainly focused on KSA DONS. Then, the characterization of database federation was explored while its strengths and weaknesses were investigated. Furthermore, some healthcare systems that use the databases federation method in their implementation were studied. Also, a description of the web services method was presented and investigation of two popular and widely used implementations of web services, Simple Object Access Protocol (SOAP)-based Web services and Representational State Transfer (REST)ful Web services, was provided as well. Finally, a review of some healthcare systems that use the web services method in their implementation was presented.

2.1 Disease Outbreak Notification Systems (DONS)

Disease outbreak detection, monitoring and notification systems play an important role in assessing threats to public health since disease outbreaks are becoming increasingly common world-wide. There are several systems in use around the world, with coverage of national, international and global disease outbreaks. The prime purpose of these systems is ensuing quick detection of possible outbreaks and epidemics.

Also, the increasing frequency of biological crises, both accidental and intentional, further illustrates the notion that Disease Outbreak Notification System (DONS) needs to be in place to meet the challenges faced by societies across the world. These systems should detect, monitor, prepare, and respond to a disease outbreak. There a large number of systems in existence that detect and prioritize potential disease outbreaks. Various DONS have been designed with different objectives, features and functions.

BioSense is an Internet-based software system that supports early detection of disease outbreaks by providing techniques for near real-time reporting, related analytics, implementation and automated outbreak detection on a national level. BioSense system collects and evaluates data from ambulatory, clinical laboratory test orders and results from Laboratory Corporation of America laboratory. It presents, summarizes, and visualizes data and analytical results through graphs, maps and tables based on day, source, state, disease type, and metropolitan area. The latest version of this application is BioSense 2.0 which provides data in a distributed cloud computing environment. [28]

The Computer Assisted Search for Epidemics (CASE) is a framework for computer supported outbreak detection. The system developed and currently in use at the Swedish Institute for Communicable Disease Control (SMI) obtains data from SmiNet and performs daily surveillance. It is open source software that removes the personal identification and includes only the specific variables in the CASE database. The system performs outbreak detection in two steps: step 1 identifies different statistical algorithms that detect unexpected or unusual number of cases from collection of patient reports for a particular disease and step 2 initiates an investigation by an epidemiologist (a human expert). If CASE detects an outbreak, step 2 aids in determining whether the detected

outbreak indicates an actual outbreak. In some cases, it might be able to detect outbreak diseases earlier than epidemiologists. Moreover, it might detect certain outbreaks that human experts would have overlooked. [29]

HealthMap is another major system that facilitates the monitoring of global infectious diseases. As described by Freifeld et al. [30], this system uses a wide variety of online formal and informal information sources and channels such as Google News, ProMED, GeoSentinel etc. to collect and aggregate content in several languages which is then classified by infectious disease agents, geography and time. The system is based on open-source products, both for its development (Linux, Apache) and its continued use (Google Maps, Google Translate API, etc). The classification mechanism is entirely automated, and is based on algorithms that use factors such as the frequency and time frame of alerts as well as the number of sources reporting the information to identify potential health events. An option is also provided for users to report outbreaks of infectious diseases in their region.

KSA DONS is an online/real-time disease outbreak notification system and designed to detect and warn potential disease outbreaks within KSA. The system notifies experts of potential disease outbreaks of both pre-listed diseases and totally unknown diseases. The system only accepts cases from pre-registered sources. It also shares information about disease outbreaks with international systems. As soon as the system detects a potential disease outbreak it notifies stakeholders and experts. The system takes feedback from experts to improve its disease detection capabilities and to adapt to new situations.

The prototype KSA DONS is implemented as a proof of concept and its implementation was a hybrid implementation using a multi-tier architecture spread across physical hosts and the private research cloud infrastructure at KFUPM. The prototype implementation was thoroughly tested for functional and technical performance with a considerably large dataset of diseases and cases. Integration testing between the various modules of KSA DONS was done as well. The prototype KSA DONS is deployed on the KFUPM Cloud Infrastructure service called KLOUD. KFUPM cloud service offers servers and storage as per required specifications from a wide range of available infrastructure templates.

The detection algorithms used in the KSA DONS were selected and adapted after a thorough study of all algorithms from selected DONS systems. KSA DONS used five efficient algorithms that are used in CASE system [29]. These five algorithms can detect isolated cases of known diseases and their potential outbreaks. Their preference for these algorithms was based on the fact that the coverage of the list of known diseases included in KSA DONS is handled by these detection algorithms. They have also implemented an outbreak detection algorithm for unknown diseases using data mining techniques. The implementation uses expert epidemiologists for consultation purposes to confirm outbreaks.

A three-tier system architecture was adapted which supports features such as scalability, availability, manageability, and resource utilization. Three-tier architecture - consisting of the presentation tier, application or business logic tier and data tier - is an architectural deployment style that describes the separation of functionality into layers with each segment being a tier that can be located on a physically separate entity. They evolved through the component-oriented approach, generally using platform specific methods for

communication instead of a message-based approach. This architecture has different usages with different applications. It can be used in web applications and distributed applications. The strength of this approach in particular is when using this architecture for geographically distributed systems. Since KSA DONS platform is cloud-based and designed to be geographically spread across the Kingdom, they were motivated to use this three tier architecture for the prototype implementation.

2.2 Database Federation

The basic idea of database federation was introduced by Hammer and McLeod [31]. A federated database is created through the integration of multiple autonomous and distributed databases into a single, logical system. Figure 2 shows a basic structure of federating system.

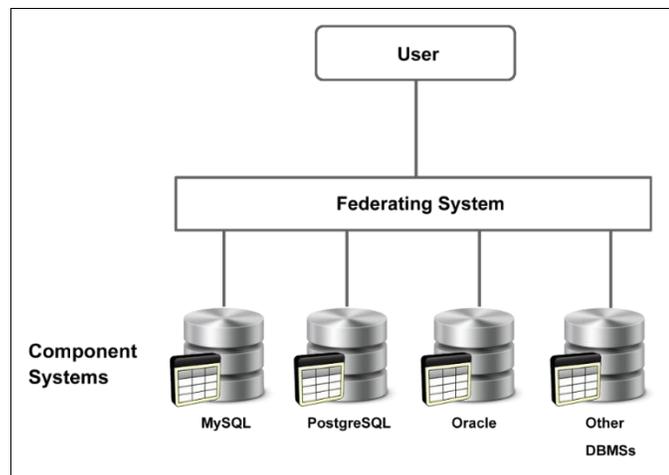


Figure 2: Basic structure of federating system

These databases can be located at different locations, have different schemas, and different semantics [32]. These differences create many problems that need to be addressed by the overall federating system.

2.2.1 Characterization of federated databases

According to [32], the components of federated databases can be characterized by autonomy, heterogeneity, and distribution.

- Autonomy is the ability of each constituent system to make a non-coerced, rational decision. It can be categorized into four types:
 - Design autonomy: each component of federated database can choose their own system data model, attributes, query language, etc. thereby, supporting heterogeneity.
 - Communication autonomy: each component of federated database can indicate when and how to communicate with other constituents of federated database.
 - Execution autonomy: local transactions/operations can be executed without influencing by operations from other components.
 - Association autonomy: each component of federated database can indicate how their operations/functions/data can be shared with other constituents of federated database.
- Heterogeneity in the data tier means that the data are not guaranteed to be located uniformly on all data sources. For example, data in one system could be in a weakly structured format, others can be in a semi-structured document or in a well-structured DBMS. There are two types of heterogeneity:
 - Structural heterogeneity means that there are differences in database schema, transaction protocol, query language, and

database model (relational and object-oriented). The global schema integration, mapping local view to a unified global schema, is the logical solution to the structural heterogeneity.

- Semantic heterogeneity means that a different meaning for the same term can exist in different component systems. Various ontologies offer an identical solution to handle the semantics heterogeneity.

Distribution means that the component databases are distributed across geographical areas, but connected by a network. This adds complexity to the federating system. However, data distribution can offer some advantages in the terms of reliability, availability, and improved access time.

2.2.2 Strengths and Weaknesses

Based on the above description, a federated database system consists of many component databases that appear to operate either as a single entity or cooperatively. Each database system that belongs to the federated system is interconnected with other components through a network of computers. Every Application Program Interface (API) that queries the components databases of the federated system will view the retrieved data as if they belong to a single distinct virtual system. Consequently, a Federated Data Base Management System (FDBMS) represents a portal between the API and the different DBMS located on different heterogeneous machines. It offers a main source of information to the public or specific users which can be then easily shared. The use of a FDBMS in the domain of disease outbreak notification systems is needed because each DONS system contains references to data in DONS systems that are located on different

geographical areas. This requires that each DONS should retrieve data from various remote data sources in the due time.

One important point that should be taken into account is how to satisfy the time requirement that is spent by the API to query and retrieve data from the virtual DB. This will create problems regarding the performance with respect to time, specifically when the remote DB tables support a massive data capacity or when the Internet speed is not reliable or slow. Basic federated system design architectures introduce a latency that cannot be accepted in real time DONS systems.

2.2.3 Federated Databases in Healthcare Systems

Many projects have implemented federated systems to provide access to their data using different integration techniques. There are many Bioinformatics systems using federated approaches such as Entrez, BioMart, and EuPathDB.

Entrez is an NCBI cross-database federated search and retrieval engine. It has a global query interface where the users can execute their queries across more than 35 databases containing over 350 million records using a single query string. The databases contained by the system are GenBank, PubMed, Nucleotide, Protein, PubMed central databases. The Entrez system provides the same simple query interface for searching each of its component databases. With the vast number of records available, Entrez contains a “limit” feature that allows the user to narrow results to a more manageable set [33]. In addition to utilizing Entrez search engine to query federated databases, NCBI provides the Entrez Programming Utilities for direct access to the results of a query. Also, it has an eUtils SOAP interface [33].

BioMart is a federated database system that provides a unified access to disparate types of database [34]. The BioMart project is based on two major models: data agnostic model and data federation. Simplifying the difficult and time consuming task of data model is done by the concept of data agnostic modelling and is achieved by using a predefined, query optimized relational schema. Organizing disparate, multiple and distributed databases into a single integrated virtual database is achieved by using the concept of data federation. BioMart allows users to use a single user interface to access and cross-reference data from multiple databases without the need for collecting the data in one location physically by database administrators [35]. BioMart was originally developed as a data warehouse for Ensemble as EnsMart. Since then EnsMart project has developed into a generic data integration package called BioMart [11]. Access is available through the BioMart Central Portal. It can be queried via several programmatic interfaces, web-based graphical user interfaces (GUIs) and third-party tools. Several integration systems are utilizing BioMart for integrating their biological data in different ways, such as CytoScape [36].

EuPathDB resources project [37] is an integrated database which include 11 databases specific to eukaryotic pathogen genomic and functional genomic data. It acts as a general portal to other specific pathogen databases such as Giardia, genera Cryptosporidium, Leishmania, Plasmodium, Trypanosoma, Trichomonas, and Toxoplasma [38][35] .

The component databases belonging to EuPathDB are integrated and assumed to be with the same graphical design and architectural structure. EuPathDB resources offers a sophisticated system for searching to enhance data accessibility that enable complex inquiry of underlying data. The system provides a builder of graphical strategy that gives

the ability to a database user to ask for complicated multi-data-type questions (queries). The recent developments in EuPathDB resources project include the design and implementation of new workflow of data loading, addition of large amounts of new data with data types and the combination of new tools for analysis [38][35][36].

2.3 Web Services

In these days, the term Web service is utilized very frequently. There are some existing popular definitions of Web services terminology ranging from comprehensive and very generic to the very restrictive and specific. One of them, it is an accessible software application over the Web to other software applications. Under this definition, almost any resources that has a Universal Resource Identifier (URI) is a Web service [41]. A formal definition introduced by IBM is that the web services are “a new breed of Web application, and they are self-contained, self-describing, modular applications that can be published, located and invoked across the Web”[42]. This definition is more comprehensive and insures that a web service has to be published, located and invoked through the Web.

The World Wide Web Consortium (W3C) defines a Web service as “a software system identified by a URI, whose public interfaces and bindings are defined and described using eXtensible Markup Language (XML). Its definition can be discovered by other software systems. These systems may then interact with the Web service in a manner prescribed by its definition, using XML based messages conveyed by Internet protocols”[43]. This definition stresses that Web services have to be well defined, described, and discovered.

Web services do program-to-program interactions, while websites do program-to-user interactions. The web services consist of a set of messaging protocols and programming standards that describe business functions and deploy them through the Internet using open standards such as Extensible Markup Language (XML), HTTP, Web Services Description Language (WSDL), Simple Object Access Protocol (SOAP), and Universal Description, Discovery and Integration (UDDI). Integration with Web Services is more easy, rapid and less expensive than ever before [44].

Normally, there are three roles included in a Web service application: service consumer, service provider, and service registry as shown in Figure 3.

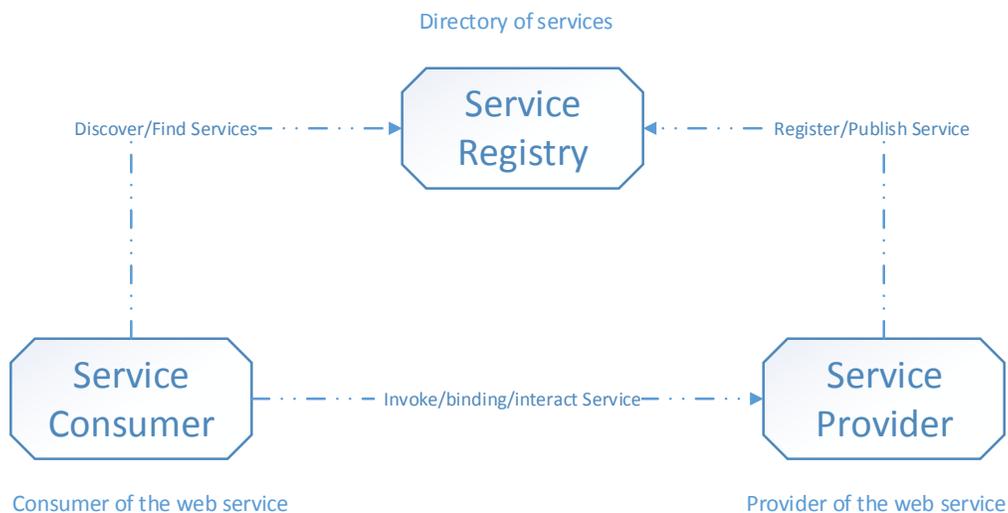


Figure 3 : The three roles in Web service application

The interactions between these three major roles are publish/register, find/discover and bind/invoke.

A service provider is responsible for implementing and publishing web services. The service provider creates the definition of the services in a WSDL and publishes it to meet

UDDI specification. A service registry is a directory of published services and it is based on the UDDI specification. The service registry offers to a service consumer a WSDL with URI to the service desired. A service consumer/requester consumes the web service and uses the registry to find information about, and access to, the Web service [45].

Currently, Web services methodology is a promising paradigm to implement service oriented architecture (SOA) and its strategic goals. A Web service is basically a set of well-defined abstraction of computational activities in semantic way involving many resources, meant to fulfill a business requirement or a customer need [41][45]. Web services are one of the primary key technologies that give power to the web as described in [46] and [47].

Recently, XML, which is adopted by Web services as a data format exchange method, has become a de-facto format standard to represent and exchange data between applications, institutions, and business processes. An XML schema is usually utilized to define the data structure of an XML document. Nowadays, there are some available robust open source software tools that can validate and parse XML documents in any platform and almost in any language in use today [48].

With the popularity of Web services technology, organizations are able to offer their internal business processes as Web services and make them available through the Internet. Currently, large companies such as Amazon, Google, Facebook, and Twitter have exposed some of their resources as Web services to be accessed by third parties to reuse and combine their services. As stated by several websites of web services publication, such as WebServiceList [49], ProgrammableWeb and WSIndex [50], Web

service technology has experienced an exponential growth in usage and popularity in the past few years. Based on recent statistics from seekda.com [41], there are 28,606 web services accessible via the web, offered by 7739 various services providers. Also, the rapid evolution of social networks, cloud computing, and Internet Web of Things accelerate increasing of web services availability on the Internet. Obviously, the web services technology will continue playing a dominant role for new software development [51][52][53][41].

Web services are offered in two most popular and widely used implementations, SOAP-based Web services and RESTful Web services. Both have been broadly accepted by the industry and academia because of the simplicity and usability. The former is based on WSDL [54] and SOAP [55], while the latter follows the REST architectural principles [56]. On the technology level, SOAP is an XML specification defining message formats and message architecture for transaction over the Internet. The SOAP message describes a top-level XML element called the envelope, which holds a header and a body. The SOAP header contains information about routing purpose and the body contains information for payload, which is described in the WSDL file [57].

The structure of a SOAP message is described by XML Schema, so at the two endpoints the SOAP engines can marshal and unmarshal its content and route it to the applicable implementation [58]. SOAP services use stack of WS-* standard protocols [59] define the interface that should be used for interaction with the other features such as service, constraint, and security.

Representational State Transfer (REST) is a lightweight infrastructure that was first introduced for building large-scale distributed hypermedia systems. The term REST is confined to only use in conjunction with HTTP. RESTful Web service is deployed in a very simple manner in building a dynamic Website [58].

WSDL is an XML language that defines the web service's interface syntactically and the reader of the service could invoke it [58]. Messages and operations are described abstractly using WSDL. This abstraction is achieved regardless of the underlying communication protocol, serialization details and the service implementation platform. A WSDL document defines web services as collections of ports, or network endpoints [60].

UDDI specifications provide users a centralized registry of services that allows them to find service providers in a unified and systematic way. Users can search, discover and download the WSDL file for the published services. The information inside the WSDL file is used to invoke the service [57]. Several standards exist to outline attributes of web services. WS-* protocol stack members such as WS-policy, WS-Transaction and WS-Addressing are used to define and advertise a service. Once you find the published web services, WS-policy discovers which best services to be added [61].

JSON-WSP (JavaScript Object Notation Web-Service Protocol) is a lightweight web service protocol using JSON (JavaScript Object Notation) for service description, requests, responses and fault. It is currently in the standardization process by the IETF (Internet Engineering Task Force) [62]. The description of JSON-WSP is based on JSON [63], which make it both human- and machine-readable. The JSON-WSP description format has the same purpose of WSDL for SOAP or IDL for CORBA, in the sense of

describing the types, nested types and methods used in a given service. Communication between a JSON-WSP server and clients are executed using HTTP POST requests and responses, with the JSON objects as data with the content-type application/ JSON [64].

Recently, the Web services have become the evolving trend to exchange data. Organizations have spent more time to provide most of their services through the Web to be accessible e.g. catalogs of their products and support for their customers. Organizations believe that having such services through web besides the traditional way will improve their efficiency in business processes [48].

External access to the systems is achieved through services which are implemented using web service technology. The choices available to use this technology is out of many reasons it is widely used and it utilizes open standards technologies and well-known protocols such as HTTP, XML, and WSDL. These open standards offer interoperability and accessibility in simple manner. Also, the web services offer functionality for accessing the databases while providing the actual implementation of the business logic of web services at the backend. The individual web services are registered in a global repository where anyone can access them. Consumers of services can access them directly from repositories, in that instance, through middleware. The consumers of a service mostly seek to utilize this service from the provided front-end web page, with which they can query for needed data [48] .

As described before, there are two major methods for implementing web services: the traditional SOAP-based Web services and conceptually simpler, RESTful Web services [58]. The following table identify the key principles, strength and weakness [65][58].

Table 1 : “SOAP-based” and “RESTful” Web services

	“SOAP-based” Web services	“RESTful” Web services
Key Principles	<ul style="list-style-type: none"> - Depend on WSDL, SOAP, and (UDDI). - Protocol independent - Usually used to integrate complex organization applications. 	<ul style="list-style-type: none"> - Utilize the REST model - Identified by URIs - Well suited for tactical, ad hoc integration over the Web
Strengths	<ul style="list-style-type: none"> - Acceptability: SOAP message format and WSDL are widespread as the gateway technologies. - Interoperability: ability to operate between heterogeneous middleware applications. - Uses HTTP and other transport protocol for transporting messages across a variety of middleware systems. - Abstraction: due to using WSDL for Service interface’s description. 	<ul style="list-style-type: none"> - Lightweight and stateless - Easy to build and test service’s client. - Deploying a RESTful Web service is like building a dynamic Website. - Thanks to URIs and hyperlinks, REST resources can be discovered without needing registration to a (centralized) repository. - Scalability, RESTful Web service serves lots of clients, thanks to the support for load balancing, caching, and clustering built into REST. - REST allows many different data

	<ul style="list-style-type: none"> - Flexibility: easily utilized in building legacy systems. - Security: SOAP supports Secure Sockets Layer (SSL), and WS-Security which adds some features to enterprise security. 	<ul style="list-style-type: none"> formats XML, JSON - Security: RESTful (Just like SOAP) supports SSL.
Weaknesses	<ul style="list-style-type: none"> - demand more computation resources, specifically when handling SOAP messages 	<ul style="list-style-type: none"> - Encoding of complex data structures into a URI may be challenge because there is no universally accepted marshalling paradigm.

From the literature, REST has become the preferred and wide-spread technology for implementing web services over time which used in web applications. SOAP-based web services, also known as WS-* Stack, is also a common alternative choice. But as we observed from the literature, there are criticisms against SOAP-based web services, specially on issues related to the complexity and bulkiness of its messages when used for web applications. Thanks to the ease of use, simplicity, and the broad use of common web-based technologies such as HTTP that the developers of web are already aware with, REST has become more widespread among developers of web applications [46][47].

The Web services method is a good solution for applications to exchange data with each other due to the fact that it uses data formats with standard protocols as mentioned earlier. Web services method is a mechanism to separate an interface from the actual

implementation. Application can describe its interface by using WSDL which is publicly accessible. The WSDL operations handle I/O parameters that are also well-defined using XML Schemas. Hence, Web services approach is more suitable for loosely coupled composition, and the provided XML schemas for each I/O parameter is easily understood by the intended applications. Integration of data between various systems is easier to achieve if the data types are similar in a structure, but it will be more difficult when they are different in the structure unless using an ontology [48].

2.4 Web Services in Healthcare Systems

Several projects have implemented web services for accessing their data, such as the National Center for Biotechnology Information (NCBI) [66], DNA Data Bank of Japan (DDJB) [67], and the European Bioinformatics Institute (EBI) [68]. We will look at two systems that implemented web services.

BioMOBY is a system used as a registry of web services in Bioinformatics that lets interoperability between biological data hosts to simplify the discovery and sharing of biological data. It operates by utilizing web services and grid technologies [69]. BioMOBY has become widely liked due to its set of standards, ensuring many systems and web services are able to interact between them. “Native BioMOBY objects are lightweight XML, and make up both the query and the response of SOAP transaction” [70]. It offers several tools to access services in its repository such as a Java API programming access [70].

Pathogen Portal (PathPort) is developed at the Virginia Bioinformatics Institute (VBI) that allows users to browse various data resources through the services deployed and

offer analysis and visualization of biological data. PathPort works together with a ToolBus, an integrated client environment, to connect to PathPort. It includes a group of web services, every one of which offers the answer to a certain query or question. They can be invoked from a WSDL file for execution without ToolBus [71].

The two systems mentioned above use web services in the Bioinformatics field. The focus of this thesis is the use of web services in disease outbreak information and notification systems to connect component databases together internally to create a unified access to a huge amount of data. To do this, we created a federated database system that supports accessible web services to be used by public as outlined in the rest of this thesis.

CHAPTER 3

THE PROPOSED FRAMEWORK

Accessing and manipulating data frequently from remote locations may cause many difficulties since the data is often stored and coded in different formats in different systems. One of the promising techniques to solve these difficulties is the federated database technique [2] that has been broadly accepted to virtualize and visualize data resident on various systems located on distributed machines. A wide range of approaches have been proposed to federate distributed databases in the literature, some of these works use cloud technology to meet privacy, security and scalability requisites, e.g., [6][7][8].

DONS Federation with web services includes providing a public portal for local databases and delivering real time answers for all the types of queries that are requested through the portal. A DONSFed (Federated system for Disease Outbreak Notification Systems) consists of a federation service and component services. The administrators of the federation system design and implement the federation services. These services divide queries into pieces and send them to the proper component systems for processing, retrieving and aggregating results return them to the end user as a whole. The component systems' administrators must support two important characteristics of their systems with regard to web services: stability and abstraction. Stability of the component system is very important because the federating system relies on it to communicate for processing

and answering the user's queries. Also, any component system should offer an abstraction of layer constituting major differences amongst the component systems to make the access consistent.

To implement our proposed framework, we need to decide on the following:

- Methodology of implementing web services integration.
- Framework layers.
- Framework workflow.

3.1 Methodology of implementing web services integration

There are several approaches to implement web services integrations to offer abstraction from the specific component systems that are part of the federation [72]. We describe some of these approaches and identify the pros and cons for each method in Table 2.

Table 2: Approaches to implement integration of web services

	Abstraction	#Services	#Operations /Services	#Queries	WSDL Discoverable	Maintenance
1	SQL statements	1	1	Many	yes	low
2	Many services	many	1	Many	yes	high
3	General service	1	many	Many	no	high
4	Categorized services with many operations	many	many	Many	yes	low

The first approach is to execute a web service as one service with a single operation. The operation would provide the connection string to the database and provide a SQL statement as input to database. This service with its operation can be described, published and discovered by a WSDL. While this method is easy to implement, it has two flaws in DONSFed design: lack of abstraction and security. In order for DONSFed system to send SQL queries to the component system, it must know the schema of the remote component databases. So, this issue will make the components systems and the DONSFed system tightly coupled with lack of autonomy. Also, if we allow any user to make a direct SQL query to be executed on the database, this will allow an attacker to inject malicious SQL statement to the database that could destroy or damage the database. Even though applying some parsing technique exist to limit this issue, it is hard to prevent it completely.

The second approach is to build one web service with one operation for each query. Each service contains a single operation related to a specific query. The service accepts parameters which are inserted to the query statement as input to retrieve requested results. As before, this service with its operation can be described, published and discovered by a WSDL. The WSDL file describes the functions of the services; and to add semantics, this service could be annotated using some Semantic Annotations for WSDL (SAWSDL) attributes. A major drawback of this approach is that it is difficult to maintain a large set of Web services. In the case of DONSFed, there are over 50 queries and when there is a need to change any parameter of a query, the corresponding service must be changed, hence increasing the maintenance concerns for both the federation administrators and the component systems.

The third approach can be a general web service with multiple operations and queries, where a single operation represents each query. This approach is similar to the previous approach and has similar pros and cons as having one web service per query. In DONSFed, there would be one web service to over than 50 operations. In addition, it is not discoverable via a WSDL and the client must have knowledge of the services' operations and the inputs to those operations to invoke this service.

The fourth approach solves the drawbacks of the third approach. It provides categorized web services with its operations and each service has its own description that describes the necessary inputs parameters that are used to invoke operations. One service is dedicated for each component system with its description and many operations for each service. Moreover, it provides many advanced features to support changes to the operations of the web service such as less maintenance effort.

3.2 Framework Layers

Our framework consists of five layers (Figure 4) namely: Interface, Query Processing, DONS Federation, Adaptation and Component Systems.

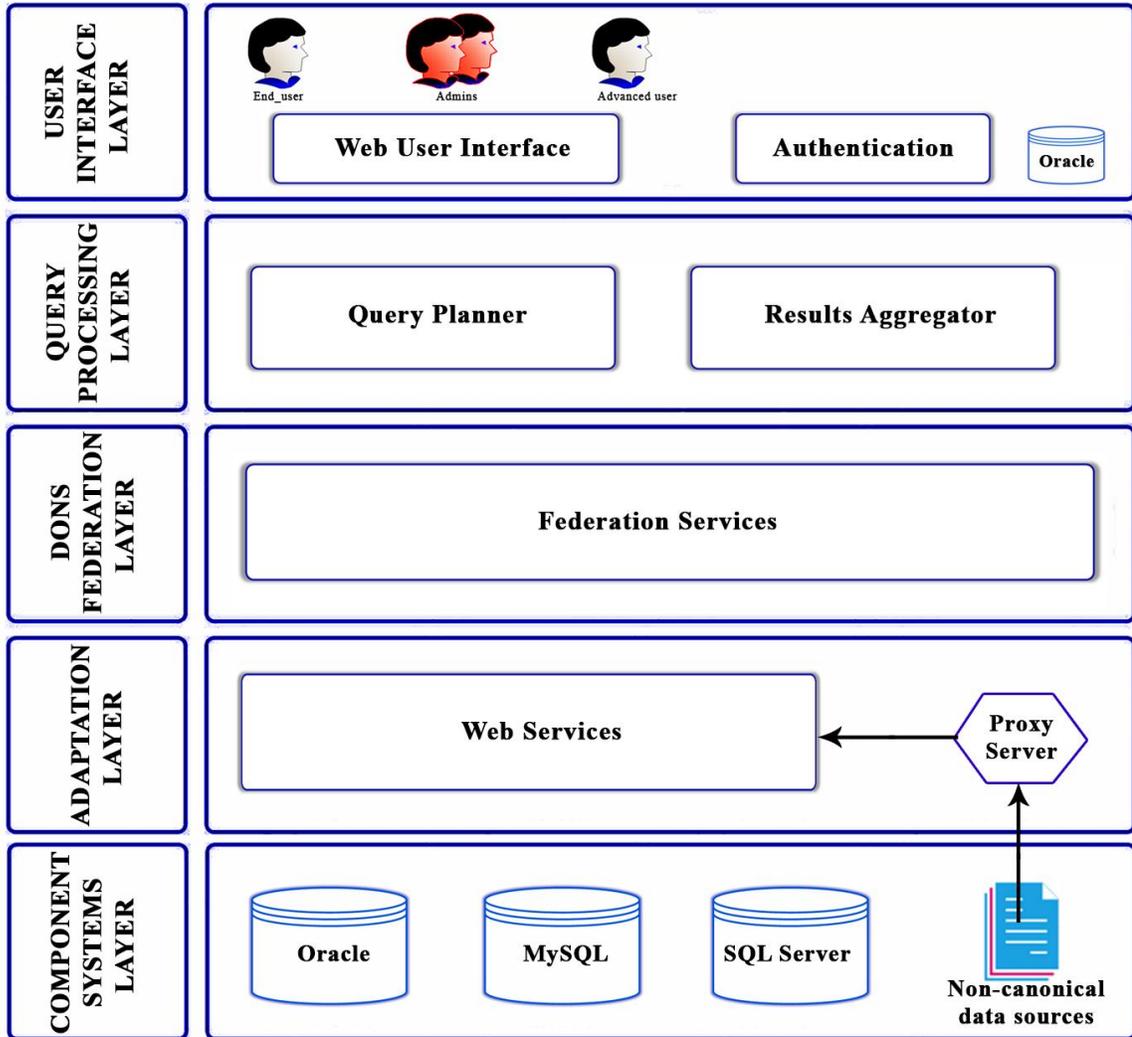


Figure 4 : Layers of DONSFed

User Interface Layer provides an interface portal for login and accessing the authentication service. The authentication service is used to verify the user and grant authorization to all subsequent federation services. The permitted data resources can be accessed based on the authenticated role of end-users, administrators or applications. Through the portal, users or applications can select from the website options that are classified into main categories, each with a set of predefined questions. Users can identify the desirable question, provide the required parameters and then submit the resulting

queries to the query service. The query service decomposes the user query into a set of sub-queries and submits them to the appropriate Web services through DONS federation layer.

In the DONS federation layer, federated services maintain the services of different database web-sites (systems) that wish to contribute to the federation. DONS federation layer is composed of several federated services. Each service is responsible for processing its own predefined requests. Based on the query, the corresponding federated service selects one (or combination) of web services that are provided by the component systems layer.

The adaptation layer maintains all Web services for each component database. It has also a proxy server to be used for non-canonical component databases - which do not support web services - to generate web services supported format.

The component systems layer consists of multiple heterogeneous data sources. Some of them have their own web services while the others, such as non-canonical data sources, are provided with proxy server to generate Web services for them. Now, it is the responsibility of component systems layer to deliver the produced information from the data sources to answer a particular request.

Once the requested data is retrieved from various data sources in XML format, the aggregate module aggregates the sub-results into a global result in a suitable format and deliver the combined results to the application or user.

The implementation of DONSFed's approach to integrate web services (Table 2, option 4) was considered. The federated services and its inputs and outputs are described

through WSDL. The queries are represented in the form of questions, most of them allow users to choose the disease or reported cases categories that they want to search and then provide the required parameters that are related to the question they select. Through the query service module, the question is translated into sub queries that use the RESTful web services to execute and get the results of the provided query. The DONS federation service should be given the URI of the required Web service of the component service and the parameters to rout the question to the proper component system.

The invocation of RESTful web services takes the provided parameters and determines which component system should be included in the invocation by DONSFed service. Once all component systems return data in an XML format, the DONS federation service parses the XML data and combines the results into a single result as an array of strings. The result is returned to the user in a nice tabular format with a respective column for each request to insure a semantically meaningful result.

The adaptation layer takes care of communication with website of DONSFed and the component systems. It picks up the required data from any component when a query requests new data.

3.3 Framework Workflow

In this section, the term workflow is defined as a set of steps that define the interactions between a user and the proposed DONSFed in order to process the user inquiries. The framework workflow is shown in Figure 5.

The proposed framework has been developed to carry out a distributed query in the DONSFed architecture in real time. Specifically, each component system belonging to

different DONS systems has a Web Service (WS) which can execute a single or multi predefined questions (from Q1 to Qn). The DONSFed portal interface contains a set of federated services that should be configured by administrators of the federated system. Each federated service contains predefined questions that can be executed by either an end user, application or administrator. Each federated service contains the instructions on how to invoke the predefined queries (Qi) to different WSs and retrieve data from such WSs. This framework can adapt to the needs of every heterogenous, distributed systems by only configuring a set of queries and the related web services.

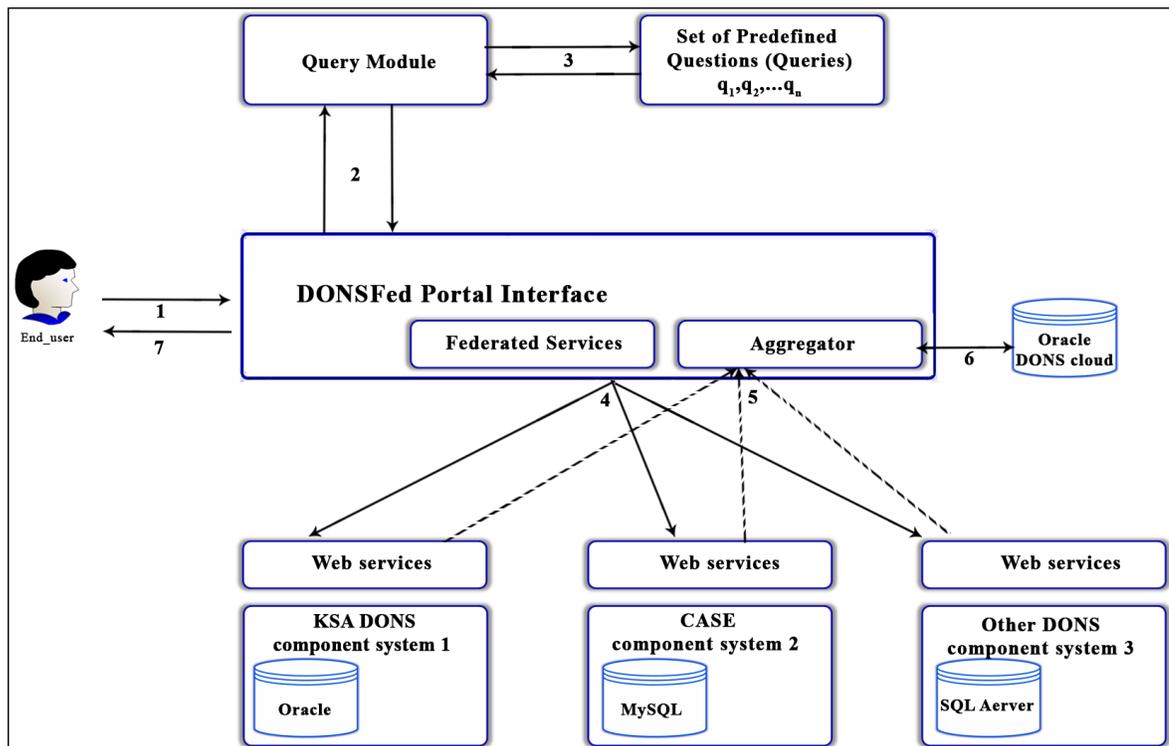


Figure 5 : Framework Workflow

Figure 5 shows an example of how our approach will work in practice: 1) the authorized user selects a specific federation service from predefined pool of federation services through the portal, 2) the federation service invokes a specific query through the query

module based on the parameters selected by the user, 3) the Query module divides the query into sub-queries and links every sub query to one of the predefined question (query) and return a batch of queries to the federation service, 4) based on the batch of queries returned by the query module, the federated service invokes a set of different WSs of each component system, 5) each web service will then process the requested query and the result from each web services will be sent to the aggregator to combine them 6) the aggregated data from different remote data sources are stored on the local server, and 7) the outcomes are shown to the user as a tabular format in an HTML page.

Figure 6 shows architectural comparisons of parallel requests with straightforward single request.

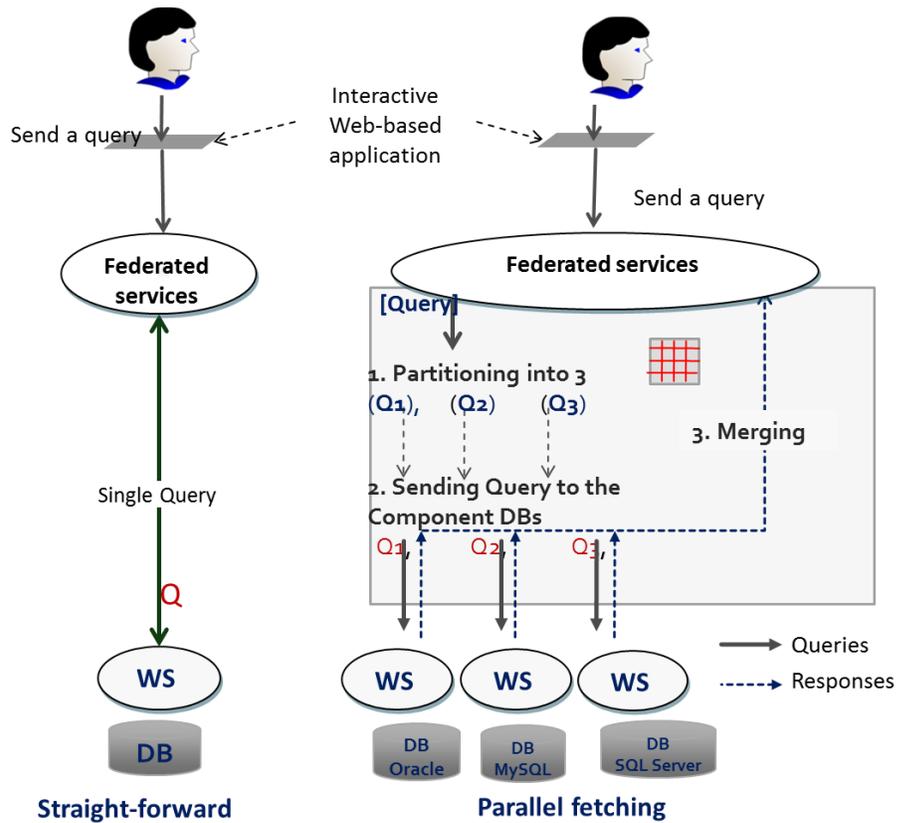


Figure 6 : Query partitioning to send each sub-query to the specific component DB

This DONSFed framework solves two major issues related to database federation. The first issue is to provide the maintenance and autonomy of the design which is very important to the component systems. DONSFed service maintains this feature by applying abstraction for all the operations occurring behind the interface of the web service. Furthermore, DONSFed service implies that changing to the services rarely occur which will require less maintenance overhead. To maintain autonomy, the DONSFed service requires no control over connected component systems.

The second issue is the heterogeneity of the component databases systems. The web services method abstracts the structural heterogeneity. Data heterogeneity is a difficult issue but in the DONSFed framework is less of a problem due to the fact that the component systems are disease outbreak notification systems yielding the uniform types of diseases, cases and outbreak data. DONSFed handles the data heterogeneity and data matching competently, so there is no need for the component systems to change anything in their data sources because they use similar naming conventions. The federated system using web services makes the DONSFed a more feasible framework.

With the federation system using web services in place, this will make it as a building block for implementing a bigger system. Once the component system has implemented web services that conform to the DONSFed, the DONSFed administrators can add other component systems to the federation with little or no effort.

3.4 Proposed Approach Features

The features of the proposed approach are:

- Using web services for implementing federation is a dynamic approach, in the sense that the components of federated system can be added and removed without degrading the overall federating system.
- Ability to access, share, and retrieve data from each component system is supported.
- By using the web services, the structure of the constituent databases can be abstracted using XML or any other semantic technique.
- Flexibility of creating a dynamic federation of databases enables maintaining and supporting autonomous component systems.
- Address the issue of heterogeneity of the distributed databases.
- Eliminates the need for local to global schema translation. In order to maintain local autonomy of constituent databases, the traditional federated database approach requires deep knowledge of local schema for each component system. This may not be feasible in some systems such as DONS where some confidential information cannot be exposed to non-authorized personals. So, our approach using Web services addressed this issue.
- Supports both compliant and non-compliant databases. In case there is a data source does not support a Web service, there is a proxy server to generate web services supported format.
- Maintains local autonomy of constituent databases.

CHAPTER 4

IMPLEMENTATION OF THE PROPOSED FRAMEWORK

4.1 Overview

Web services provide solutions to three common issues of federation which are distribution, heterogeneity and autonomy. Distribution issue is handled by the availability of services over the web using well-known standard protocols. Structural heterogeneity and autonomy are handled by applying abstraction to the component data sources that web services offer. Web services can provide an abstraction of the underlying data sources and thus there is no need for low-level knowledge of the underlying component systems. Semantic heterogeneity is handled by common terms or ontology designed by federation service administrators. Agreement on predefined common terms ensures that data from disparate component data sources can be joined by the associated federation service into a single meaningful result.

The basic principle of federated system is that we can integrate several related systems without modifying or discarding the component systems. And no need for current management to abandon full control of their system. The implementation of web services for federated system should allow communication to achieve these goals. Actually web services can offer both the robustness and autonomy needed for the federated system. Also, web services allow for component systems to be added to the federated system with

no modifications whatsoever. The local administrators of component databases systems must implement web services that adhere to the federation system. Once this is complete, the local administrators are responsible for describing the logic for the implemented web services in order to query and get results from their system. This logic is very important to ease the addition of repositories to the federated system with minimal effort by the administrators of federation system and without modifying the component systems.

Further, one can think of the web services as remote procedure calls. In this regard, the web service represents the application and its operations represent the procedure. The inputs to the web service are well-defined in the WSDL definitions to be of a specific type and structure and they are similar to parameters of a procedure. Also, the outputs defined in the WSDL definitions are similar to the return value of the invoked procedure. Within this analogy, we can say that it is easy for a web service to provide high-level of abstraction of the federated system to which it is offering access. Due to this high-level abstraction, the developers of the component systems keep full control over their system's implementation.

Also, web services provide another form of autonomy. The component system must design and implement its service so that it is accessible by the federated system, but other technical details about the service's implementation are left up to the developers of the component system. Since RESTful web services use HTTP methods and XML standards to communicate, there are no constraints on the programming language. Thus each component system keeps total autonomy over its services and data while it is included in the federated system.

Web services offer other benefits over other techniques of federation implementations. Utilizing Web services for the purpose of federation builds a dynamic federation due to the fact that component systems can come and go without corrupting the federation. This dynamic federation quality improves a great deal of federation's functionality. First, suppose that a component system could not be available to respond to a query due to a failure. This component system can leave the respective federated system without leaving any gaps in the functionality, but of course affecting the availability of data, while the federated system will still provide its functionality.

Web services provide another benefit if the federation service is published; they will be accessible from outside the federated system. If the federated service is published, the related component services get published and are accessible. A federation is like an access point which is known to other component systems and the component systems as the nodes, these nodes do not know about each other but they can communicate with each other through the access point. Publishing federation services gives each node in the federation the ability to send and retrieve data from any of the other nodes. Web service federation allows the nodes to leave the federation and return back without any extra works and these results in a truly dynamic federation. Any candidate system can also easily join the federation.

4.2 Development Environment Setup

In this section, we describe prototype implementation architecture, databases and data service servers that are used to implement a proof-of-concept implementation.

4.2.1 Prototype Implementation Architecture

Since our prototype is a cloud-based and geographically spread across multiple heterogeneous platforms, we were motivated to use a three-tier architecture for our prototype implementation.

The presentation tier is the topmost level of the application. The presentation layer provides the application's user interface. Typically, this involves the use of Graphical User Interface for smart client interaction, and Web based technologies for browser-based interaction. As shown in the Figure 7 , the terminals for primary health centres, experts and any healthcare practitioners use the DONSFed browser-based application for data entry, data collection, data integration and decision making. The external databases such as WHO and others are also connected through this layer for data transmission and retrieval.

The logic tier controls an application's functionality by performing detailed processing. Logic tier is where mission-critical business problems are solved. The components that make up this layer can exist on a server machine, to assist in resource sharing. These components can be used to enforce business rules, such as business algorithms and data rules, which are designed to keep the data structures consistent within either specific or multiple databases. Because these middle-tier components are not tied to a specific client, they can be used by all applications and can be moved to different locations, as response time and other rules required. In Figure 7, the web servers and application server constitutes the logic tier. A cluster of web servers and applications servers can be used for load balancing and failover among the cluster nodes.

The data tier consisting of database servers is the actual DBMS access layer. It can be accessed through the business services layer and on occasion by the user services layer. Here information is stored and retrieved. This tier keeps data neutral and independent from application servers or business logic. Giving data its own tier also improves scalability and performance.

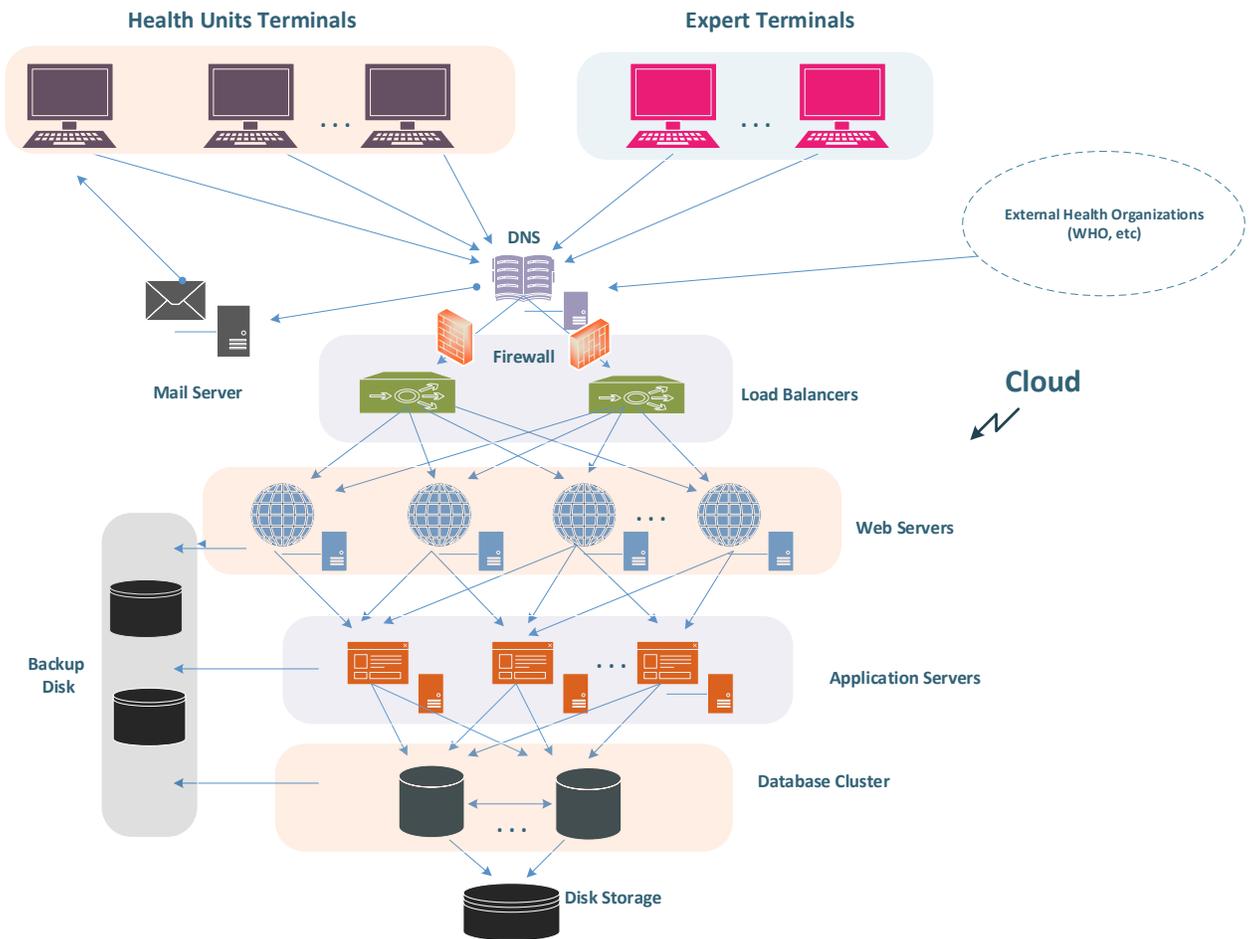


Figure 7 : DONSFed Prototype System Architecture

The flow of data in the three tiered architecture is described next. In the presentation layer, the users access the DONSFed applications over the network through the web browser. The request is securely sent over the network to a firewall. Then the trusted

requests from the firewall are forwarded to load balancers. The firewall ensures that trust relationships between the presentation and application tiers are complied with. The trusted requests are sent to a DNS server for name resolution and to load balancers which are capable of distributing the load across the web servers and manage the network traffic. In the logic tier of the DONSFed, web and application servers are deployed to handle all user requests. While the user requests (http or https) are served by the Apache web servers, the application servers handle all the business logic processing and data processing. The data tier provides all the data needed for the logic tier through SQL queries using database specific protocol over TCP/IP. The database tables are maintained by insertion, updating and deleting of data. To avoid loss of data due to data corruption or system failure, the data backup of all the critical servers and databases is performed periodically in a separate storage location. The data retention policy is applied for timely recovery of data in case of any disaster.

Figure 8 shows the deployment diagram, which visualizes the hardware, the middleware and the software used in our proof-of-concept implementation. The diagram consists of multiple tiers: web servers, clients, data sources, and integration application tiers. Different tiers are communicated together via several interfaces and protocols. Incoming HTTP requests from the users are first go through the DNS server. Then web balancer server forwards requests to different web servers. Web servers interact with the application server to handle the requests and get the appropriate response. The response will be obtained from different component systems each one is hosted in different virtual machines in the cloud using web services middleware.

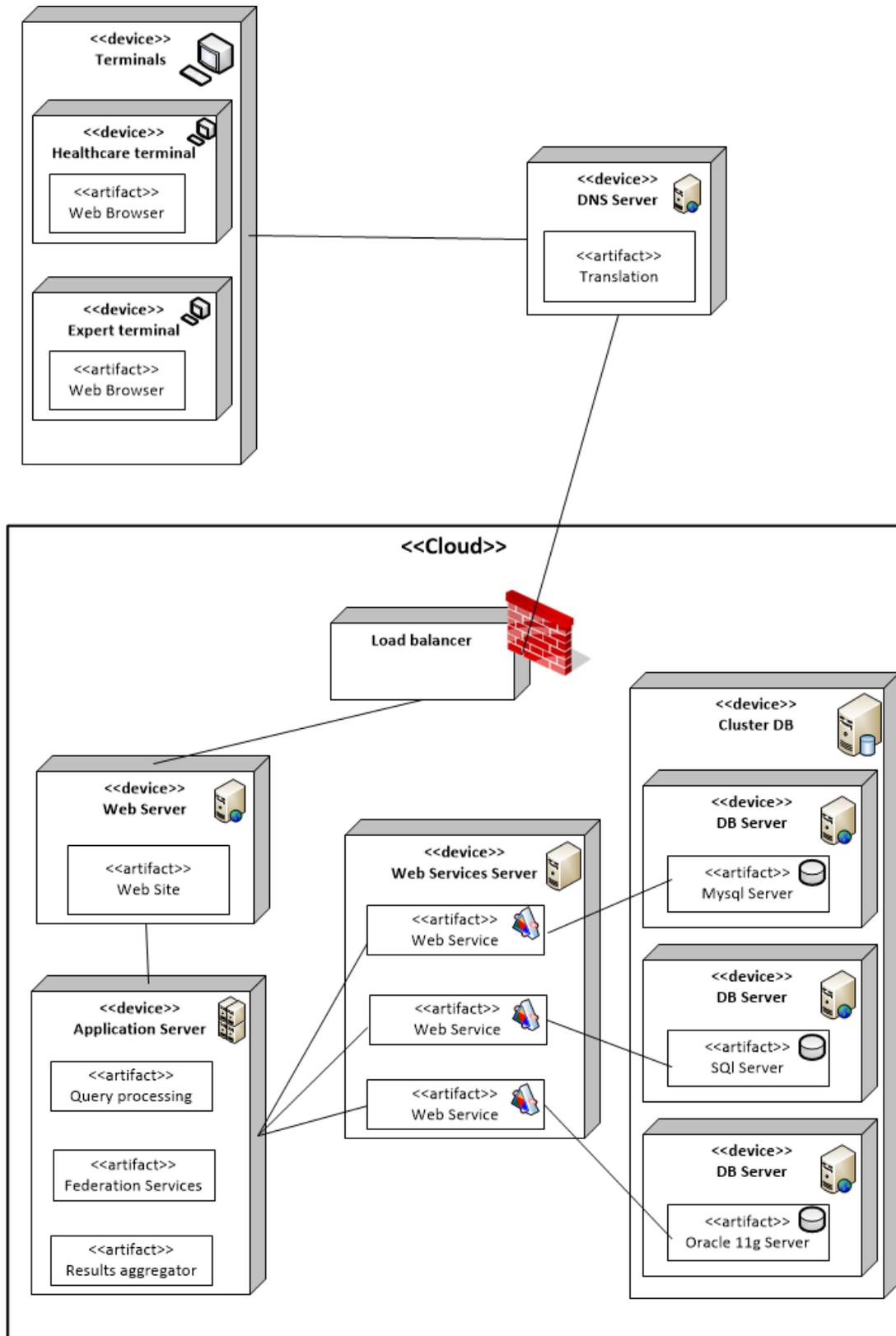


Figure 8 : Deployment diagram of DONSFed

4.2.2 Databases

For providing proof-of-concept implementation to the proposed framework, we used three different autonomous and distributed databases. These databases are located at different locations, have different schemas, and different semantics, which are suitable for testing our federation framework.

The first database is part of the KSA DONS system which is an Oracle cloud-based database. The KSA DONS database server is a KLOUD virtual machine with Red Hat Linux 6.4 as its operating system. The Oracle client software was configured on all the servers and clients in the KSA DONS architecture in order to communicate with each other and to connect to the database Server. The complete KSA DONS database schema on the Oracle platform is shown in Figure 9. The KSA DONS database schema consists of 19 tables, along with stored procedures, triggers and views.

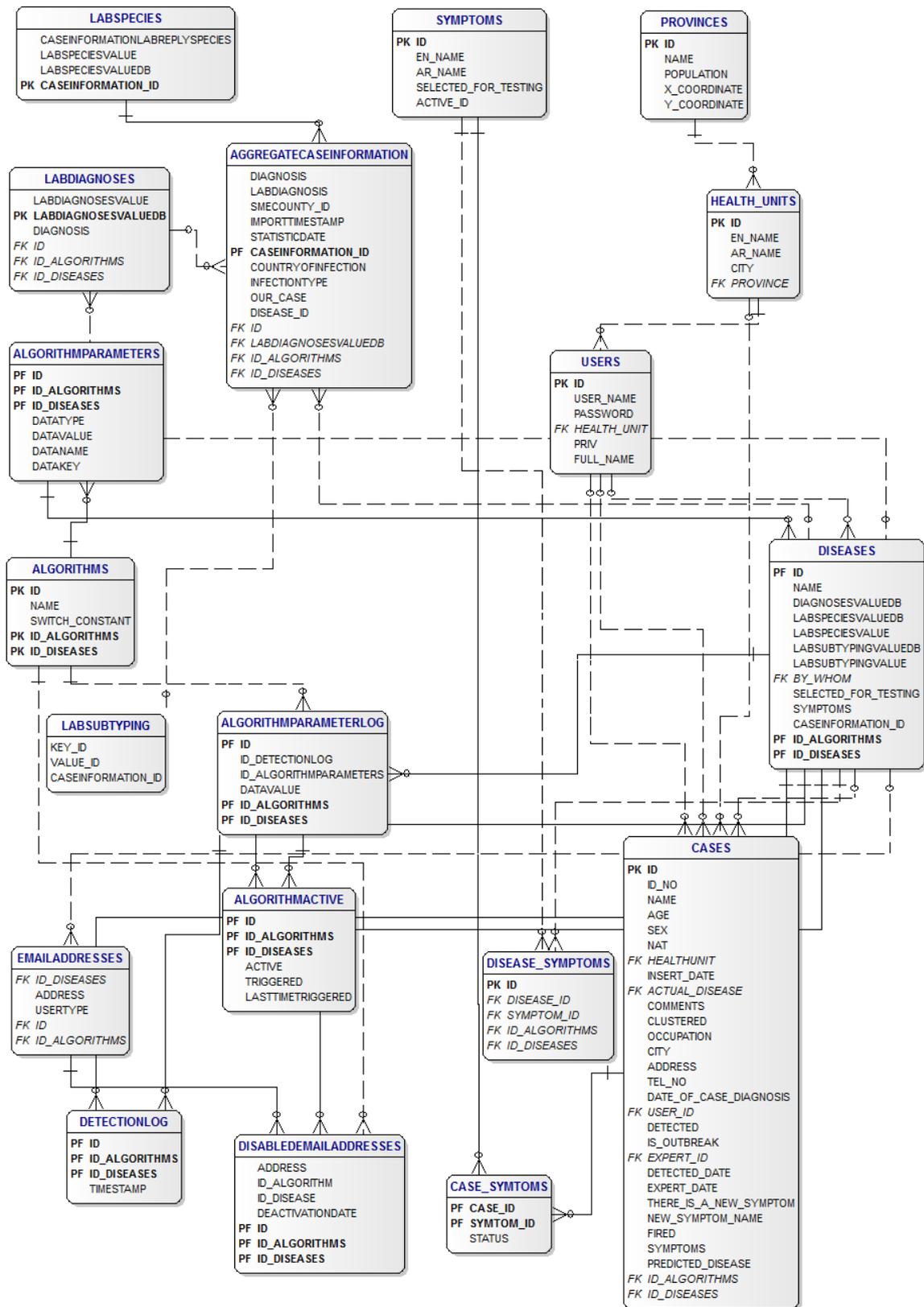


Figure 9 : Complete KSA DONS database schema on the Oracle platform

The second database was acquired from the CASE system which is a MySQL database. The CASE system was developed and is currently in use at the Swedish Institute for Communicable Disease Control (SMI). The system obtains data from the national notifiable disease database in Sweden (SmiNet) and performs daily surveillance. It is an open source software that removes the personal identification and includes only specific variables from the CASE database. The complete CASE database schema on the MySQL platform is shown in Figure 10.

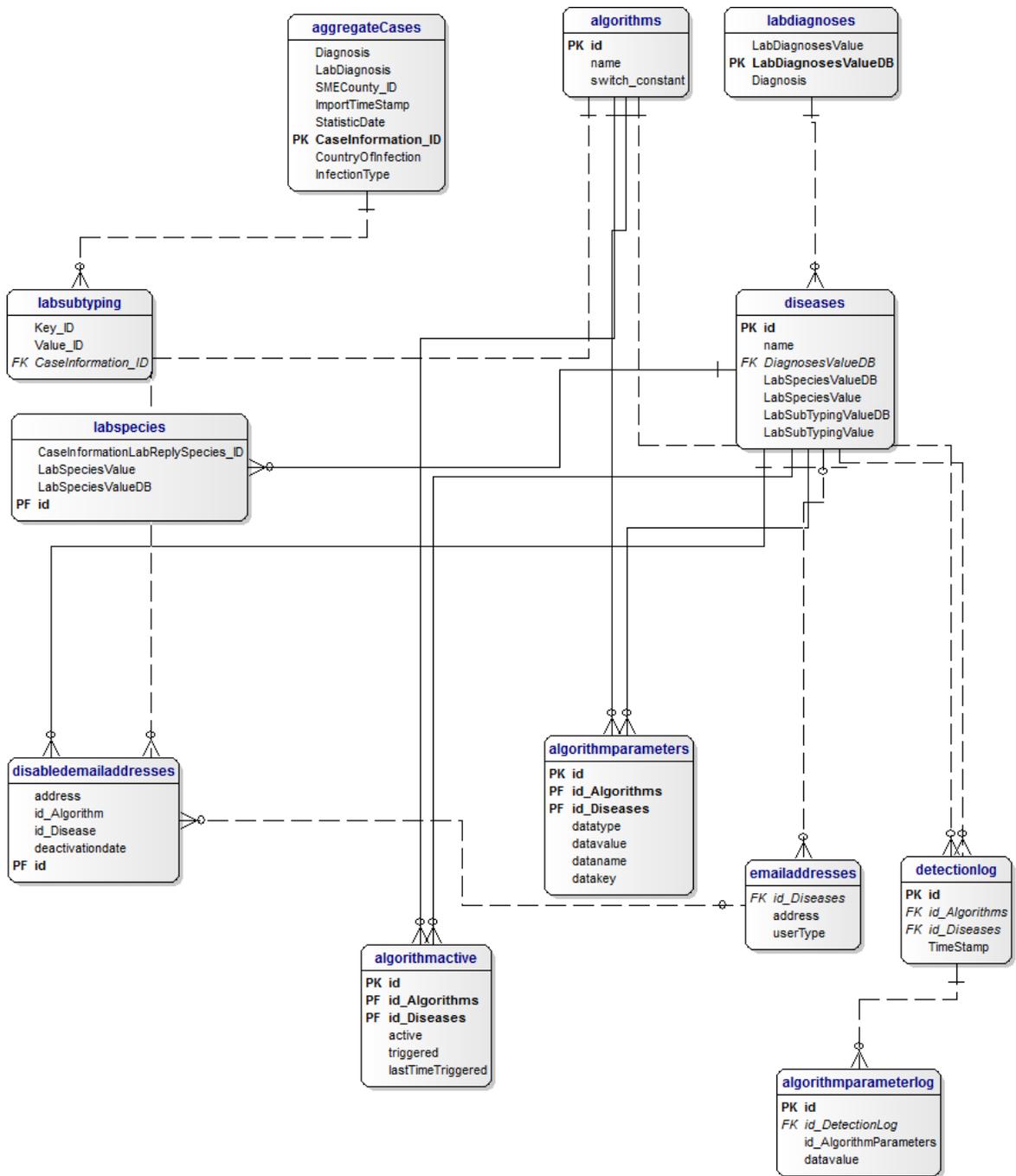


Figure 10 : Complete CASE database schema on the MySQL platform

To implement and validate our prototype to build a federated database using web services, we needed at least three databases. So, we created the third database by going through two phases of development. Initially, the third database was designed with

MySQL platform and configured for use in the CASE system. Subsequently, the entire database with all the associated objects such as the database schema, stored procedures, triggers etc., was migrated to the SQL Server database platform in order to provide more heterogeneity. The database server is configured with Microsoft SQL Server Express Version 2013 relational DBMS and the database was created using SQL Server Migration Assistant (SSMA) utility offered by Microsoft SQL Server.

The SSMA has built-in migration support to migrate database objects and data from databases like MySQL, Oracle, Microsoft Access and Sybase. Before the actual migration process starts, the environment for migration has to be prepared. This involves: setting up the project-level options to convert objects, migrate data, and map source data types to target data types and ensure that all configuration options are suitable for our framework. We chose the Optimistic mode for configuration options because it keeps more of the current MySQL syntax.

After connecting to MySQL and SQL Server, we mapped MySQL Database to SQL Server Schemas (MySQLToSQLSERVER). Then, we converted the MySQL database to SQL Server and synchronized the database objects. Finally, we migrated all MySQL data into SQL Server database. The database schema is where we store the metadata and it consists of 12 tables, 8 views, the primary keys and indexes required, along with the stored procedures and triggers. Figure 11 show the connection established between MySQL and SQL Server with the SSMA tool.

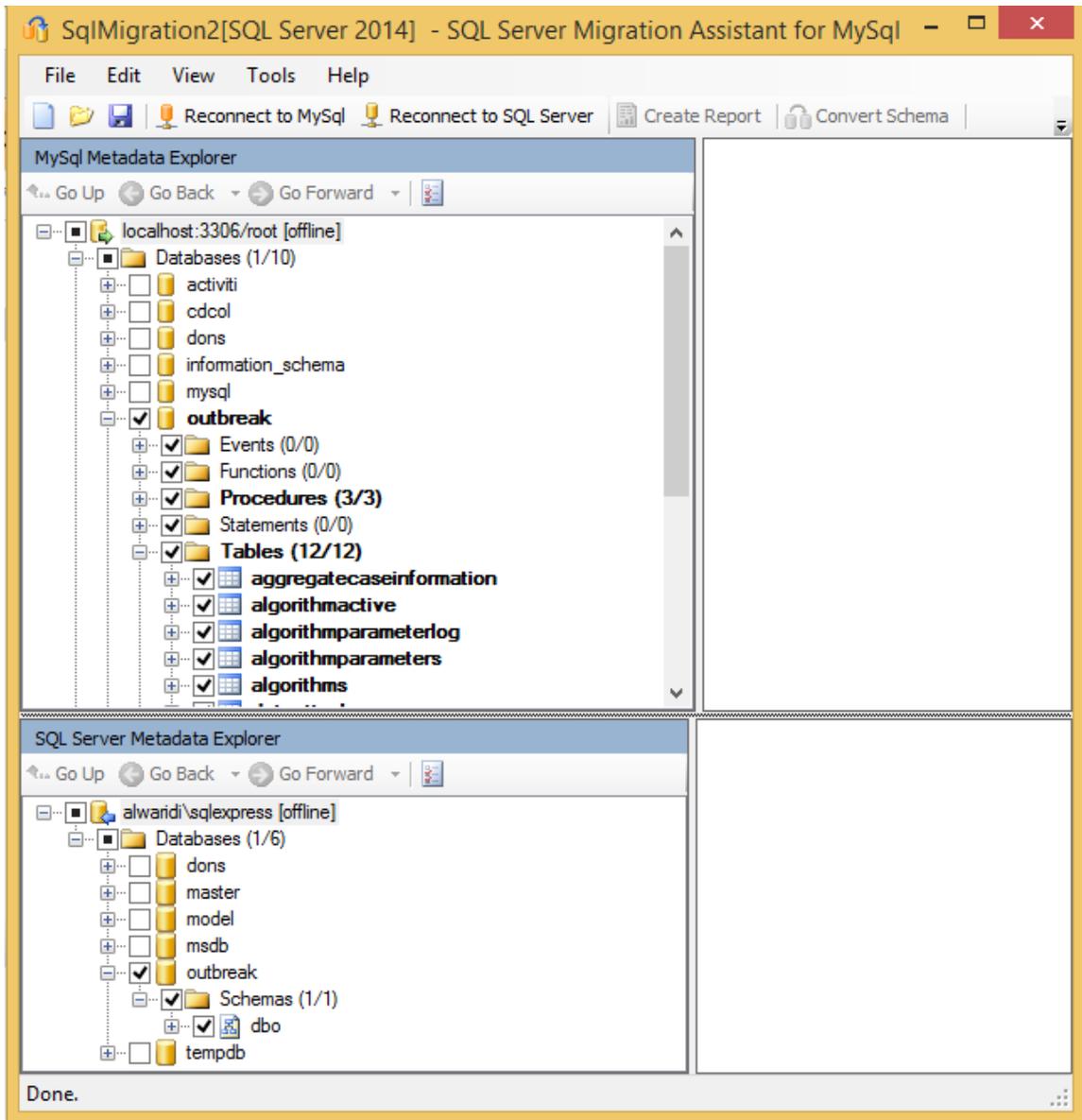


Figure 11 : MySQL database migration to SQL Server using SSMA tool

The complete DONS database schema on the SQL Server platform is shown in Figure 12.

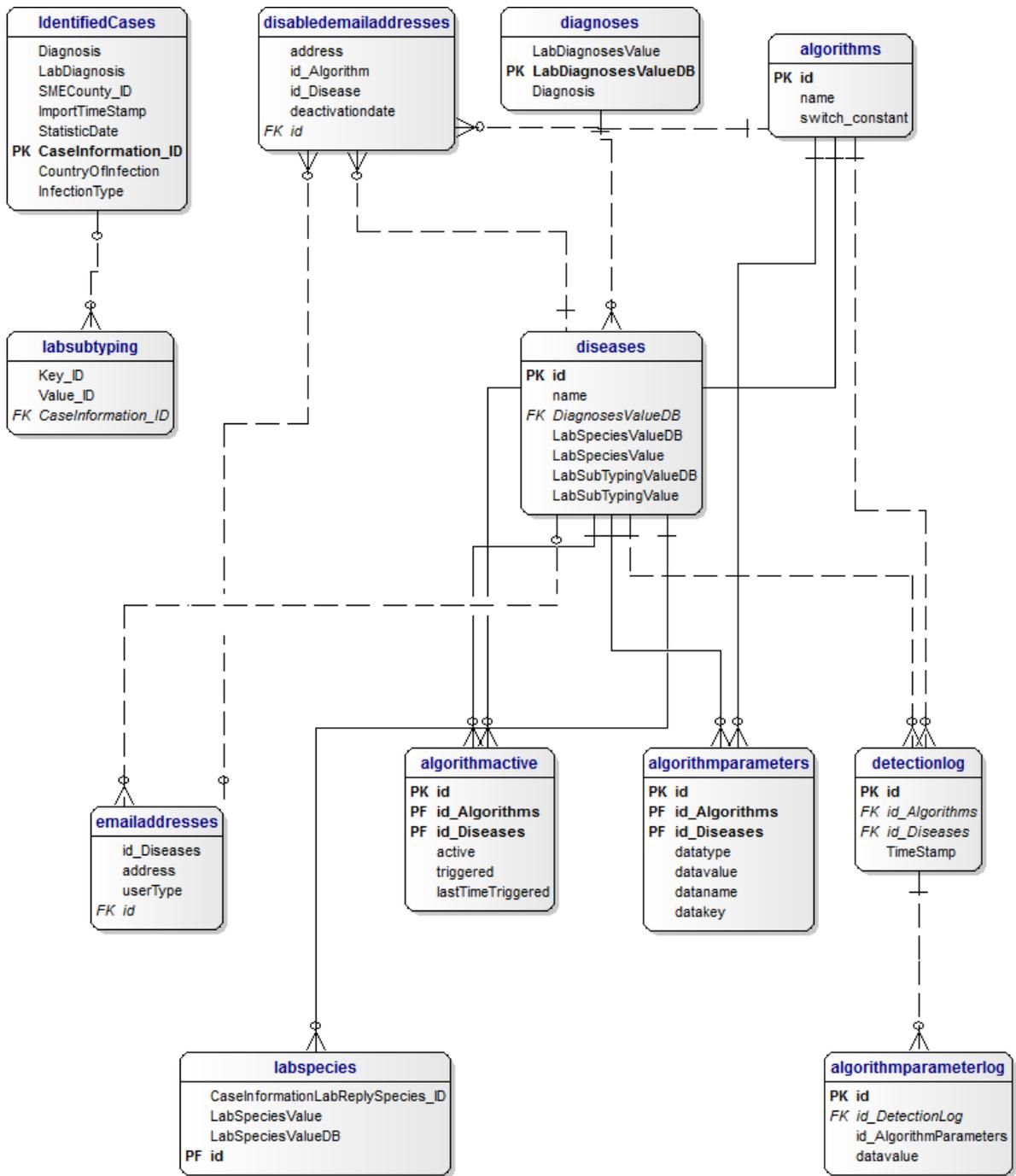


Figure 12 : Complete DONS database schema on the SQL Server platform

4.2.3 Data Services Server

In this thesis, we implemented web services using a data services server (adaptation layer) that utilizes the principle of service-oriented architecture to offer uniform access to autonomous and heterogeneous data sources. This approach ensures that the data service layer is used to mask heterogeneity between databases, spreadsheets, or files that represent the data sources, and makes them accessible as web services such as the RESTful web service.

The main advantage of this approach is that it reduces the complexity of building new applications that access and retrieve data from multiple data sources. This approach also saves the time required in the development of new applications by supporting the integration of multiple existing data sources through federation service. [73].

A 'Data Service' is simply a mechanism to take data located away in heterogeneous databases and other file formats such as XML and Excel sheets and present them as SOAP_based Web Services or as RESTful style Web resources. There are some benefits of data services such as exposing relational databases (e.g. JDBC, MySQL, Oracle, SQL Server or DB2 in addition to CSV, Excel and LDAP files) as XML based web services. Also, they map a single file for exposing data into various different formats and protocols. This benefit will cut down the redundant programming required to access various different database technologies. Further, they reflect the changes that happen to the core data through all channels, without having to discard an entire data set. This leads to reduced bottleneck of network bandwidth. They also expose data as both REST-style Web resources and SOAP-based (WS-* style) Web services for consumption. Data can also be exposed as JSON supporting exposure of various, different data sources as a

unified resource. Additionally, they generate unified views and support transparency of a steady repository that affords better accessibility to organization data and support highly effective maintenance of data stores. Furthermore, they provide loose-coupling of data with consuming APIs [74]. We implemented the data services server using WSO2 data services server [75]. This server provides an easy to use platform to integrate data stores, create composite data views, and host data services.

4.2.4 Experiments

In this section, we provide some of the experimental results of our implementation of federation services. Figure 13 shows a web service called the DONS Federation service connected to Oracle database (DONSFedOracle web service) which consists of many operations.

We used the “TryIt” tool as an alternate https which is provided by WSO2 data services to test the web service. The left side of the Figure 13 shows all operations related to the DONSFedOracle web service. The service description is found in the top of Figure 13. When we select one operation of the DONSFedOracle web service, the request statement appears in XML format in the request window. Then, we provide its parameters and press the SEND button. The results will appear in XML format in the response window as shown in Figure 13.

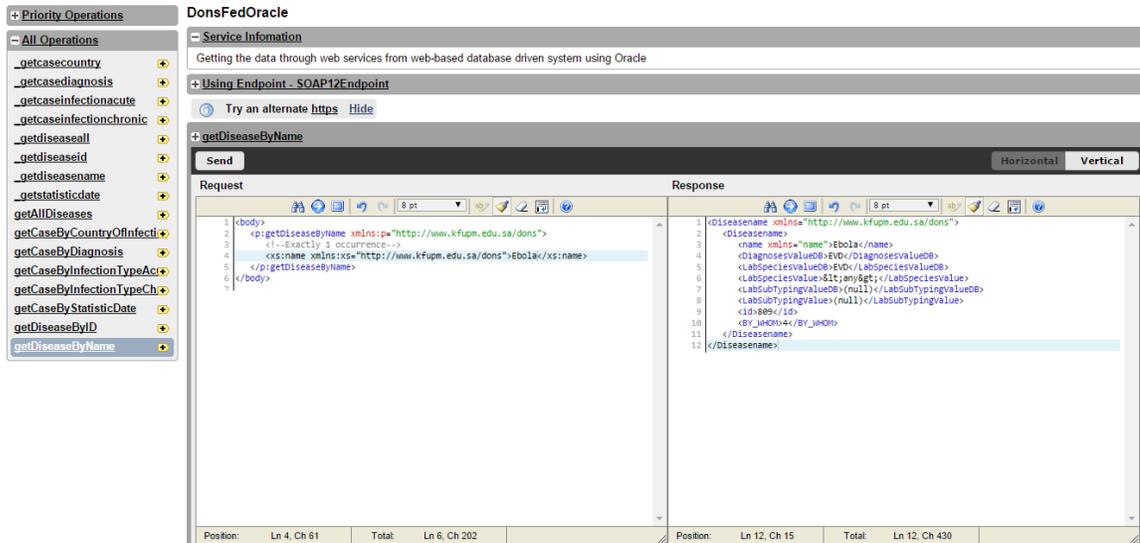


Figure 13 : DONSFedOracle data service

Figure 13 also shows a web service with its operations that provides accessibility to data in a single component database system which is hosted on the Oracle platform. Here, each operation of the web service answers a predefined query and specifies the parameters which are needed to provide the accessibility to the data. The results from the executed operation is shown in XML format. Also, it provides the service information that describes the service with its operation. Further, it provides the WSDL file to this service.

For example, we can select one operation called “getDiseaseByName” which takes the name as input parameter of the desired disease to search and collects information about it. We provided an “Ebola” disease as input parameter to the getDiseaseByName operation and received the response in the right window which has all information about the Ebola disease. The WSDL file for this service in the next section provides further details of this transaction.

Figure 14 shows another web service, DONS Federation service connected to MySQL (DONSFedMySQL web service) which also consists of many operations. We used the “TryIt” tool to test the DONSFedMySQL web service. The left side of the Figure 14 shows all operations that relate to the DONSFedMySQL web service. We selected one operation called “getCaseByStatisticDate” which will provide us with all the identified cases (diagnosed cases) and their information such as diagnosis, country of infection and infection type between two dates. This operation takes two input parameters, the start date and the end date (time duration). We provided two different dates as input parameters to the getCaseByStatisticDate operation and we received the response in the response window as shown in Figure 14 which lists identified cases and its corresponding information.

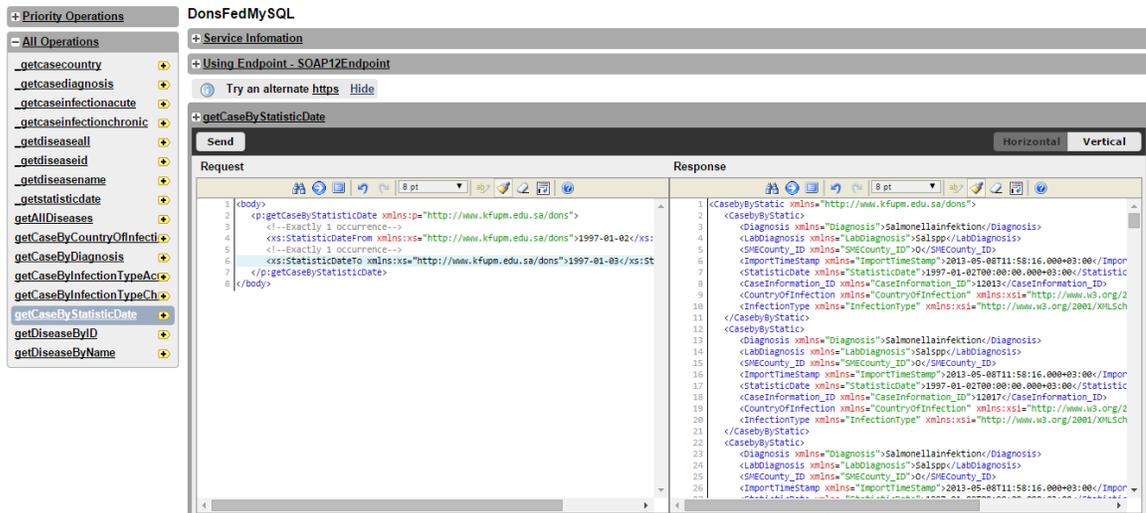


Figure 14 : DONSFedMySql data service

Another web service, DONS Federation service which is connected to SQLServer (“DONSFedSQLServer” web service) is shown in Figure 15 and it again consists of many operations. We used the “TryIt” tool to test it. The left side of the Figure 15 shows

all operations that related to the DONSFedSQLServer web service. In Figure 15, we selected an operation called “getCaseByCountryOfInfection” which will provide us with all the identified cases (diagnosed cases) which are related to a specific country with additional information such as diagnosis and infection type. This operation takes the country of infection as input parameter. We provided “Togo” country as country of infection parameter to the getCaseByCountryOfInfection operation and we received the response in the response window listing identified cases and the requested information related to the country of Togo.

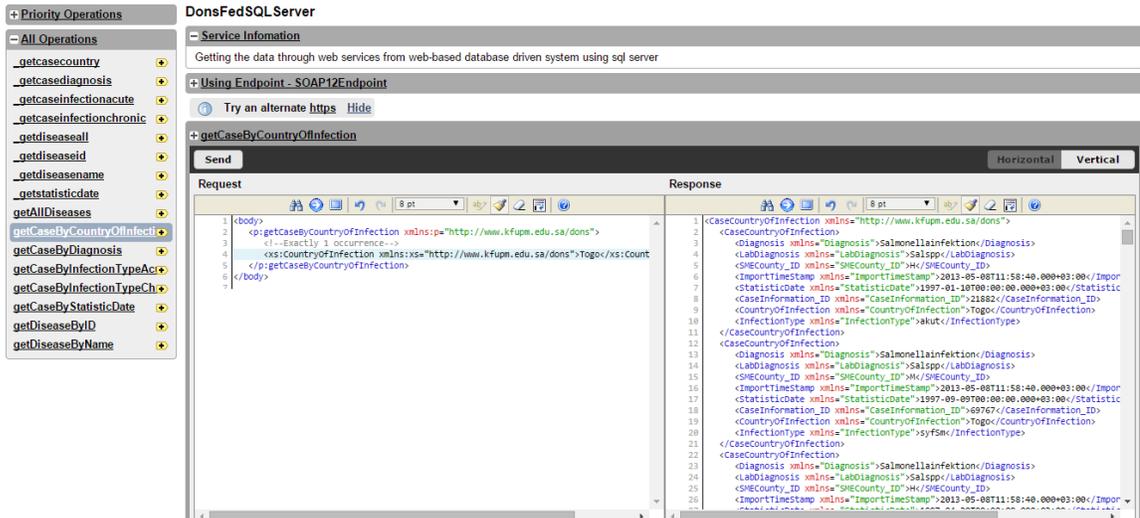


Figure 15 : DONSFedSQLServer data service

As demonstrated in the previous examples, the DONSFed homepage offers easy and quick access for the user by providing links to specific component databases sites. The left side of the DONSFed homepage (Figure 16) offers an expandable window which includes descriptions, news, community resources, upcoming events, and additional information. The middle of the DONSFed homepage includes links to access the federation services that aggregate all data available in component databases not restricted

to particular component database. The DONSFed user has the ability to search data that can be found on any component database.

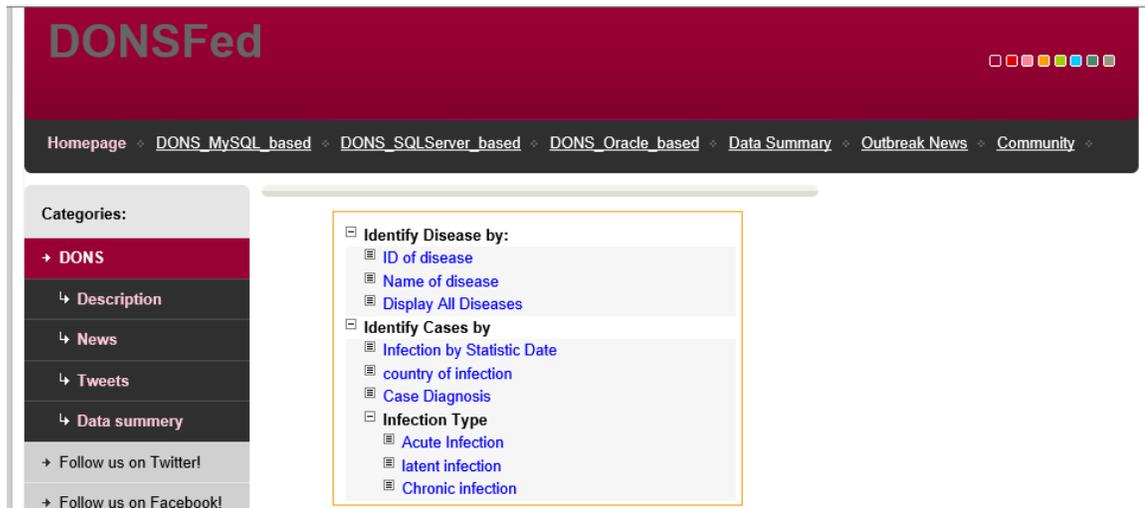


Figure 16 : DONSFed home page

From the results of all component databases, the user can determine which disease is an outbreak in a particular time and in a specific location based on the registered cases. Also, the user can generate a comparable list of all registered cases identify the required data. Execution the DONSFed queries starts by selecting from the two categories that specify the type of search based on the disease or registered cases.

The first category represents the identified diseases by ID, name, and the user can expose all diseases that are related to one specific component database or to all component databases. Then, the user has to provide the required parameters for that web service and send the query to receive the results.

The second category represents the identified cases by the following associations: date of infection, country of infection, diagnosis of case and by the type of infection such as

acute, chronic, and latent infection. Each of these associations has its own required parameters that must be provided to the web services to obtain the necessary results.

Homepage ◊ DONS_MySQL_based ◊ DONS_SQLServer_based ◊ DONS_Oracle_based ◊ Data Summary ◊ Outbreak News ◊ Community ◊

Categories:

- DONS
- ↳ Description
- ↳ News
- ↳ Tweets
- ↳ Data summary
- Follow us on Twitter!
- Follow us on Facebook!
- Follow us on YouTube!
- Download Community Files
- Upcoming Events
- Related Sites
- About DONSFed

RSS feeds:

- Articles
- Comments

Please insert the start date and end date then press search
 From: To:

Total	DONSFed1 SQL_System1	DONSFed2 SQLServer_System2	DONSFed3 Oracle_System3
96 rows	31 rows	28 rows	37 rows

Diagnosis	LabDiagnosis	SMECounty_ID	ImportTimeStamp	StatisticDate	CaseInformation_ID	CountryC
Ebola	EVD	AC	2014-05-13T00:00:00.000+03:00	2014-04-06T00:00:00.000+03:00	20008	Uganda
Ebola	EVD	AC	2014-06-13T00:00:00.000+03:00	2014-05-06T00:00:00.000+03:00	20009	Uganda
Ebola	EVD	AC	2013-06-13T00:00:00.000+03:00	2013-05-06T00:00:00.000+03:00	20019	Congo
Salmonellainfektion	Salspp	H	2014-10-24T11:46:44.000+03:00	2013-05-06T00:00:00.000+03:00	3730551	Bosnien
Salmonellainfektion	Salspp	H	2013-05-08T11:58:15.000+03:00	2013-05-06T00:00:00.000+03:00	3730677	
Syphilis	Tpalli	H	2013-05-08T11:58:15.000+03:00	2013-05-06T00:00:00.000+03:00	3730718	
Salmonellainfektion	Salspp	M	2013-05-08T11:58:39.000+03:00	2013-05-06T00:00:00.000+03:00	3730893	Armenien
Salmonellainfektion	Salspp	M	2014-10-24T11:46:44.000+03:00	2013-05-06T00:00:00.000+03:00	3730957	Armenien
Salmonellainfektion	Salspp	H	2013-05-08T11:58:15.000+03:00	2013-05-06T00:00:00.000+03:00	3731270	
VRE	VREfum	H	2013-05-08T11:58:15.000+03:00	2013-05-06T00:00:00.000+03:00	3731422	

Figure 17 : Federated service for case identification by statistics date

All the results are dynamically inserted into datasets that are presented in tables. Figure 17 shows one federation service that searches the registered cases on all component databases based on the statistic dates duration from a start date to specific an end date. These dates are inserted as parameters to the requested service. The aggregator service module parses the XML response and the results are shown initially in a generic table that specifies how many cases found in each of the component database and the total results from all component databases. Furthermore, from the generic table the user can click on the hyperlinks to explore the data for each component database or for all the data simultaneously. As shown in Figure 17, the results of the query from the first component system based on MySQL database are 31 cases, from the second component system

based on SQL Server database are 28 cases, and from the last component system based on Oracle database are 37 cases. All results from the component databases are integrating to one table by clicking on the 'Total' hyperlink.

Figure 18 shows another federation service that searches for a specific registered disease on all component databases based on the name of a disease that is inserted as a parameter to the requested service "getDiseaseByName". The aggregator service module parses the XML response and the results are shown initially in a generic table that specifies how many diseases matched the inserted name in each of the component database and presents the total results from all component databases. Furthermore, from the generic table the user can click on the links to explore the data for each or for all the data simultaneously. As shown in Figure 18, we inserted "SARS" disease as input parameter to the getDiseaseByName federation service. The results of the query from the first component system based on MySQL database are 2 names matched the SARS disease, from the second component system based on SQL Server database are 2 names matched the SARS disease, and from the last component system based on Oracle database is only 1 name. All results from the component databases are integrating into one table by clicking on the 'Total' hyperlink.

Homepage ◊ DONS MySQL based ◊ DONS SQLServer based ◊ DONS Oracle based ◊ Data Summary ◊ Outbreak News ◊ Community ◊

Categories:

- + DONS
- ↳ Description
- ↳ News
- ↳ Tweets
- ↳ Data summary
- Follow us on Twitter!

Please insert the disease name and press search

sars

Total	DONSFed1 SQL_System1	DONSFed2 SQL Server_System2	DONSFed3 Oracle_System3
5 rows	2 rows	2 rows	1 rows

ID	Name	DiagnosesValueDB	LabSpeciesValueDB	LabSpeciesValue	LabSubTypingValueDB	LabSubTypingValue
49	SARS	SARS	<any>	<any>		
768	SARS	SARS	SARS	SARS coronavirus		

Figure 18 : Federated service for disease identification by disease name

Homepage ◊ DONS MySQL based ◊ DONS SQLServer based ◊ DONS Oracle based ◊ Data Summary ◊ Outbreak News ◊ Community ◊

Categories:

- + DONS
- ↳ Description
- ↳ News
- ↳ Tweets
- ↳ Data summary
- Follow us on Twitter!
- Follow us on Facebook!
- Follow us on YouTube!
- Download Community Files
- Upcoming Events
- Related Sites
- About DONSFed

RSS feeds:

- Articles
- Comments

Get all cases with Chronic Infection Type

Total	DONSFed1 SQL_System1	DONSFed2 SQL Server_System2	DONSFed3 Oracle_System3
177 rows	71 rows	53 rows	53 rows

DiagnosisSMECounty_ID	ImportTimeStamp	StatisticDate	CaseInformation_ID	CountryOfInfection	InfectionType
H	2014-10-24T11:48:19.000+03:00	2004-09-15T00:00:00.000+03:00	645	Colombia	chron
M	2014-10-24T11:48:19.000+03:00	2004-09-15T00:00:00.000+03:00	1135	Armenien	chron
Ispp	2014-10-24T11:48:19.000+03:00	2005-06-30T00:00:00.000+03:00	927478	Sverige	chron
Ispp	2014-10-24T11:48:19.000+03:00	2005-07-04T00:00:00.000+03:00	929591	Sverige	chron
Ispp	2014-10-24T11:48:19.000+03:00	2005-10-03T00:00:00.000+03:00	993362	Colombia	chron
Ispp	2014-10-24T11:48:19.000+03:00	2005-10-11T00:00:00.000+03:00	998021	Italien	chron
Ispp	2014-10-24T11:48:19.000+03:00	2005-12-19T00:00:00.000+03:00	1037095	Guatemala	chron
Ispp	2014-10-24T11:48:19.000+03:00	2005-12-28T00:00:00.000+03:00	1040550		chron
Ispp	2014-10-24T11:48:19.000+03:00	2006-01-10	1044354	Rosnien	chron

Figure 19 : Federated service for getting all Cases which their infection type is Chronic

Figure 19 shows another federation service that searches for registered cases which their infection types are chronic through all component databases. The requested service “getCaseByInfectionTypeChronic” dose not have parameters. Once we click on the search button to collect all cases which are chronic, we receive the results and the aggregator service module parses the XML response and the results are shown initially in

a generic table that specifies how many cases matched the infection type with chronic in each of the component database and presents the total results from all component databases. Furthermore, from the generic table the user can click on the hyperlinks to explore the data for each component database or for all the data simultaneously. The results of the query from the first component system based on MySQL database are 71 chronic cases, from the second component system based on SQL Server database are 53 chronic cases, and from the last component system based on Oracle database is 53 chronic cases. All results from the component databases are integrating into one table by clicking on the ‘Total’ hyperlink.

Homepage ◊ DONS MySQL based ◊ DONS SQLServer based ◊ DONS Oracle based ◊ Data Summary ◊ Outbreak News ◊ Community ◊

Categories:

- DONS
- ↳ Description
- ↳ News
- ↳ Tweets
- ↳ Data summary
- Follow us on Twitter!
- Follow us on Facebook!
- Follow us on YouTube!
- Download Community Files
- Upcoming Events
- Related Sites
- About DONSFed

RSS feeds:

- Articles

Please insert the Country of infection and press search

Togo Search

Total	DONSFed1 SQL_System1	DONSFed2 SQLServer_System2	DONSFed3 Oracle_System3
331 rows	106 rows	106 rows	119 rows

Diagnosis	LabDiagnosis	SMECounty_ID	ImportTimeStamp	StatisticDate	CaseInformation_ID	Country
Salmonellainfektion	Salspp	H	2014-10-24T11:48:44.000+03:00	1997-01-10T00:00:00.000+03:00	21882	Togo
Salmonellainfektion	Salspp	M	2013-05-08T11:58:40.000+03:00	1997-09-09T00:00:00.000+03:00	69787	Togo
Salmonellainfektion	Salspp	H	2014-10-24T11:48:44.000+03:00	1997-04-30T00:00:00.000+03:00	81808	Togo
Salmonellainfektion	Salspp	N	2014-10-24T11:48:44.000+03:00	1997-12-17T00:00:00.000+03:00	109029	Togo
Salmonellainfektion	Salspp	H	2013-05-08T11:58:40.000+03:00	1998-01-14T00:00:00.000+03:00	121889	Togo
Salmonellainfektion	Salspp	H	2014-10-24T11:48:44.000+03:00	1998-07-16T00:00:00.000+03:00	156817	Togo
Salmonellainfektion	Salspp	M	2014-10-24T11:48:44.000+03:00	1998-07-10T00:00:00.000+03:00	157293	Togo
Salmonellainfektion	Salspp	H	2014-10-24T11:48:44.000+03:00	1998-08-19T00:00:00.000+03:00	166429	Togo

Figure 20 : Federated service for case identification by the country of infection

Figure 20 shows another federation service that searches for specific registered cases on all component databases based on the country of infection. The country of infection is inserted as a parameter to the requested service “getCaseByCountryOfInfection”. As

shown in Figure 20, we inserted “Togo” as a country of infection parameter to get all related cases to the `getCaseByCountryOfInfection` federation service. The results of the query from the first component system based on MySQL database are 106 cases, from the second component system based on SQL Server database are 106 cases, and from the last component system based on Oracle database are 119 cases. All results from the component databases are integrating into one table by clicking on the ‘Total’ hyperlink.

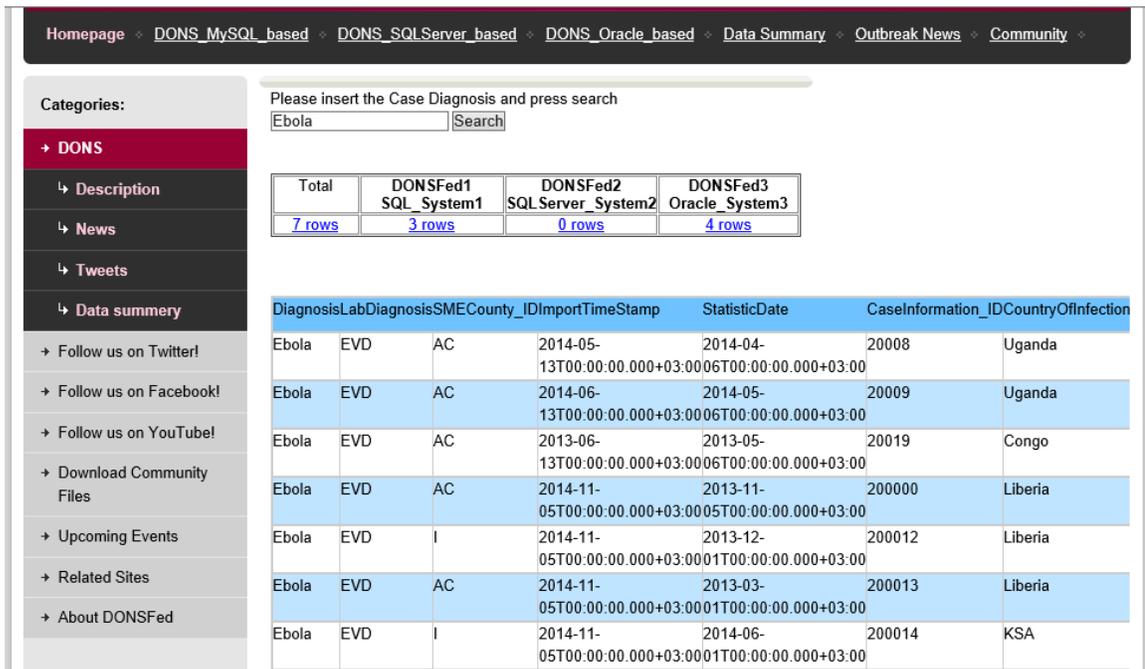


Figure 21 : Federated service for case identification by its diagnosis

Figure 21 shows another federation service that searches for specific registered cases on all component databases based on the case diagnosis. The diagnosis is inserted as a parameter to the requested service “`getCaseByDiagnosis`”. As shown in Figure 21, we inserted “Ebola” as a diagnosis parameter to get all related cases to the `getCaseByDiagnosis` federation service. The results of the query from the first component system based on MySQL database are 3 cases, from the second component system based on SQL Server database are 0 cases, and from the last component system based on Oracle

database are 4 cases. All results from the component databases are integrating into one table by clicking on the ‘Total’ hyperlink. All other federation services are implemented in the same way of the aforementioned examples and are ready to be utilized by the users of our DONSFed website.

4.2.5 An Example of data service

In this section we provide an example of data service implementation for one of our component database that is based on Oracle platform.

```
<data enableBatchRequests="true" name="DonsFedOracle"
serviceNamespace="http://www.kfupm.edu.sa/dons">
  <description>      Getting the data through web services from web-based database
driven system using Oracle
</description>
  <config id="DonsOracle">
    <property name="driverClassName">oracle.jdbc.driver.OracleDriver</property>
    <property
name="url">jdbc:oracle:thin:outbreak/outbreak@dons.kfupm.edu.sa:1521/orallg.kfupm.edu.sa<
/property>
    <property name="username">outbreak</property>
    <property name="password">outbreak</property>
  </config>
  <query id="DiseaseByid" useConfig="DonsOracle">
    <sql>SELECT * FROM diseases WHERE id =?</sql>
    <result element="diseaseID" rowName="diseaseID">
      <element column="id" name="id" namespace="id" xsdType="integer"/>
      <element column="name" name="name" namespace="name" xsdType="string"/>
      <element column="DIAGNOSESVALUEDB" name="DIAGNOSESVALUEDB"
namespace="DIAGNOSESVALUEDB" xsdType="string"/>
      <element column="LABSPECIESVALUE" name="LABSPECIESVALUE"
namespace="LABSPECIESVALUE" xsdType="string"/>
      <element column="LABSPECIESVALUEDB" name="LABSPECIESVALUEDB"
namespace="LABSPECIESVALUEDB" xsdType="string"/>
      <element column="LABSUBTYPINGVALUE" name="LABSUBTYPINGVALUE"
namespace="LABSUBTYPINGVALUE" xsdType="string"/>
      <element column="LABSUBTYPINGVALUEDB" name="LABSUBTYPINGVALUEDB"
namespace="LABSUBTYPINGVALUEDB" xsdType="string"/>
    </result>
    <param name="id" sqlType="INTEGER"/>
  </query>
  <query id="DiseaseByName" useConfig="DonsOracle">
    <sql>SELECT * FROM diseases where name=?</sql>
    <result element="Diseasename" rowName="Diseasename">
      <element column="name" name="name" namespace="name" xsdType="string"/>
      <element column="DiagnosesValueDB" name="DiagnosesValueDB" xsdType="string"/>
      <element column="LabSpeciesValueDB" name="LabSpeciesValueDB" xsdType="string"/>
      <element column="LabSpeciesValue" name="LabSpeciesValue" xsdType="string"/>
      <element column="LabSubTypingValueDB" name="LabSubTypingValueDB"
xsdType="string"/>
      <element column="LabSubTypingValue" name="LabSubTypingValue" xsdType="string"/>
      <element column="id" name="id" xsdType="integer"/>
      <element column="BY_WHOM" name="BY_WHOM" xsdType="integer"/>
    </result>
    <param name="name" sqlType="STRING"/>
  </query>
  <query id="DiseaseAll" useConfig="DonsOracle">
```

```

    <sql>select * from diseases</sql>
    <result element="AllDiseases" rowName="AllDiseases">
      <element column="id" name="id" namespace="id" xsdType="string"/>
      <element column="name" name="name" namespace="name" xsdType="string"/>
      <element column="DIAGNOSESVALUEDB" name="DIAGNOSESVALUEDB"
namespace="DIAGNOSESVALUEDB" xsdType="string"/>
      <element column="LABSPECIESVALUE" name="LABSPECIESVALUE"
namespace="LABSPECIESVALUE" xsdType="string"/>
      <element column="LABSPECIESVALUEDB" name="LABSPECIESVALUEDB"
namespace="LABSPECIESVALUEDB" xsdType="string"/>
      <element column="LABSUBTYPINGVALUE" name="LABSUBTYPINGVALUE"
namespace="LABSUBTYPINGVALUE" xsdType="string"/>
      <element column="LABSUBTYPINGVALUEDB" name="LABSUBTYPINGVALUEDB"
namespace="LABSUBTYPINGVALUEDB" xsdType="string"/>
    </result>
  </query>
  <query id="CasebyDiagnosis" useConfig="DonsOracle">
    <sql>SELECT * FROM aggregatecaseinformation where Diagnosis=?</sql>
    <result element="CaseDiagnosis" rowName="CaseDiagnosis">
      <element column="Diagnosis" name="Diagnosis" namespace="Diagnosis"
xsdType="string"/>
      <element column="LabDiagnosis" name="LabDiagnosis" namespace="LabDiagnosis"
xsdType="string"/>
      <element column="SMECounty_ID" name="SMECounty_ID" namespace="SMECounty_ID"
xsdType="string"/>
      <element column="ImportTimeStamp" name="ImportTimeStamp"
namespace="ImportTimeStamp" xsdType="dateTime"/>
      <element column="StatisticDate" name="StatisticDate" namespace="StatisticDate"
xsdType="dateTime"/>
      <element column="CaseInformation_ID" name="CaseInformation_ID"
namespace="CaseInformation_ID" xsdType="integer"/>
      <element column="CountryOfInfection" name="CountryOfInfection"
namespace="CountryOfInfection" xsdType="string"/>
      <element column="InfectionType" name="InfectionType" namespace="InfectionType"
xsdType="string"/>
    </result>
    <param name="Diagnosis" sqlType="STRING"/>
  </query>
  <query id="CasebyCountryOfInfection" useConfig="DonsOracle">
    <sql>SELECT * FROM aggregatecaseinformation WHERE CountryOfInfection=?</sql>
    <result element="CaseCountryOfInfection" rowName="CaseCountryOfInfection">
      <element column="Diagnosis" name="Diagnosis" namespace="Diagnosis"
xsdType="string"/>
      <element column="LabDiagnosis" name="LabDiagnosis" namespace="LabDiagnosis"
xsdType="string"/>
      <element column="SMECounty_ID" name="SMECounty_ID" namespace="SMECounty_ID"
xsdType="string"/>
      <element column="ImportTimeStamp" name="ImportTimeStamp"
namespace="ImportTimeStamp" xsdType="dateTime"/>
      <element column="StatisticDate" name="StatisticDate" namespace="StatisticDate"
xsdType="dateTime"/>
      <element column="CaseInformation_ID" name="CaseInformation_ID"
namespace="CaseInformation_ID" xsdType="integer"/>
      <element column="CountryOfInfection" name="CountryOfInfection"
namespace="CountryOfInfection" xsdType="string"/>
      <element column="InfectionType" name="InfectionType" namespace="InfectionType"
xsdType="string"/>
    </result>
    <param name="CountryOfInfection" sqlType="STRING"/>
  </query>
  <query id="CasebyInfectionTypeAcute" useConfig="DonsOracle">
    <sql>SELECT * FROM aggregatecaseinformation WHERE InfectionType = 'acute'</sql>
    <result element="CaseInfectionTypeAcute" rowName="CaseInfectionTypeAcute">
      <element column="Diagnosis" name="Diagnosis" namespace="Diagnosis"
xsdType="string"/>
      <element column="LabDiagnosis" name="LabDiagnosis" namespace="LabDiagnosis"
xsdType="string"/>
      <element column="SMECounty_ID" name="SMECounty_ID" namespace="SMECounty_ID"
xsdType="string"/>
      <element column="ImportTimeStamp" name="ImportTimeStamp"
namespace="ImportTimeStamp" xsdType="dateTime"/>

```

```

        <element column="StatisticDate" name="StatisticDate" namespace="StatisticDate"
xsdType="dateTime"/>
        <element column="CaseInformation_ID" name="CaseInformation_ID"
namespace="CaseInformation_ID" xsdType="integer"/>
        <element column="CountryOfInfection" name="CountryOfInfection"
namespace="CountryOfInfection" xsdType="string"/>
        <element column="InfectionType" name="InfectionType" namespace="InfectionType"
xsdType="string"/>
    </result>
</query>
<query id="CasebyInfectionTypeChronic" useConfig="DonsOracle">
    <sql>SELECT * FROM aggregatecaseinformation WHERE InfectionType='chron'</sql>
    <result element="CaseInfectionTypeChronic" rowName="CaseInfectionTypeChronic">
        <element column="Diagnosis" name="Diagnosis" namespace="Diagnosis"
xsdType="string"/>
        <element column="LabDiagnosis" name="LabDiagnosis" namespace="LabDiagnosis"
xsdType="string"/>
        <element column="SMECounty_ID" name="SMECounty_ID" namespace="SMECounty_ID"
xsdType="string"/>
        <element column="ImportTimeStamp" name="ImportTimeStamp"
namespace="ImportTimeStamp" xsdType="dateTime"/>
        <element column="StatisticDate" name="StatisticDate" namespace="StatisticDate"
xsdType="dateTime"/>
        <element column="CaseInformation_ID" name="CaseInformation_ID"
namespace="CaseInformation_ID" xsdType="integer"/>
        <element column="CountryOfInfection" name="CountryOfInfection"
namespace="CountryOfInfection" xsdType="string"/>
        <element column="InfectionType" name="InfectionType" namespace="InfectionType"
xsdType="string"/>
    </result>
</query>
<query id="CasebyByStatic" useConfig="DonsOracle">
    <sql>SELECT * FROM aggregatecaseinformation where StatisticDate between ? and
?</sql>
    <result element="CasebyByStatic" rowName="CasebyByStatic">
        <element column="Diagnosis" name="Diagnosis" namespace="Diagnosis"
xsdType="string"/>
        <element column="LabDiagnosis" name="LabDiagnosis" namespace="LabDiagnosis"
xsdType="string"/>
        <element column="SMECounty_ID" name="SMECounty_ID" namespace="SMECounty_ID"
xsdType="string"/>
        <element column="ImportTimeStamp" name="ImportTimeStamp"
namespace="ImportTimeStamp" xsdType="dateTime"/>
        <element column="StatisticDate" name="StatisticDate" namespace="StatisticDate"
xsdType="dateTime"/>
        <element column="CaseInformation_ID" name="CaseInformation_ID"
namespace="CaseInformation_ID" xsdType="integer"/>
        <element column="CountryOfInfection" name="CountryOfInfection"
namespace="CountryOfInfection" xsdType="string"/>
        <element column="InfectionType" name="InfectionType" namespace="InfectionType"
xsdType="string"/>
    </result>
    <param name="StatisticDateFrom" sqlType="DATE"/>
    <param name="StatisticDateTo" sqlType="DATE"/>
</query>
<operation name="getDiseaseByID">
    <description>
        get Disease By ID
    </description>
    <call-query href="DiseaseByid">
        <with-param name="id" query-param="id"/>
    </call-query>
</operation>
<operation name="getDiseaseByName">
    <call-query href="DiseaseByName">
        <with-param name="name" query-param="name"/>
    </call-query>
</operation>
<operation name="getCaseByDiagnosis">
    <description>
        get Case By Diagnosis
    </description>
    <call-query href="CasebyDiagnosis">

```

```

        <with-param name="Diagnosis" query-param="Diagnosis"/>
    </call-query>
</operation>
<operation name="getAllDiseases">
    <description>        get All Diseases
</description>
    <call-query href="DiseaseAll"/>
</operation>
<operation name="getCaseByInfectionTypeAcute">
    <description>        get Case By Infection Type Acute
</description>
    <call-query href="CasebyInfectionTypeAcute"/>
</operation>
<operation name="getCaseByInfectionTypeChronic">
    <description>        get Case By Infection Type Chronic
</description>
    <call-query href="CasebyInfectionTypeChronic"/>
</operation>
<operation name="getCaseByCountryOfInfection">
    <description>        get Case By Country Of Infection
</description>
    <call-query href="CasebyCountryOfInfection">
        <with-param name="CountryOfInfection" query-param="CountryOfInfection"/>
    </call-query>
</operation>
<operation name="getCaseByStatisticDate">
    <description>        get Case By Statistic Date
</description>
    <call-query href="CasebyByStatic">
        <with-param name="StatisticDateFrom" query-param="StatisticDateFrom"/>
        <with-param name="StatisticDateTo" query-param="StatisticDateTo"/>
    </call-query>
</operation>
<resource method="GET" path="diseaseid">
    <call-query href="DiseaseById">
        <with-param name="id" query-param="id"/>
    </call-query>
</resource>
<resource method="GET" path="diseasename">
    <call-query href="DiseaseByName">
        <with-param name="name" query-param="name"/>
    </call-query>
</resource>
<resource method="GET" path="diseaseall">
    <call-query href="DiseaseAll"/>
</resource>
<resource method="GET" path="casediagnosis">
    <call-query href="CasebyDiagnosis">
        <with-param name="Diagnosis" query-param="Diagnosis"/>
    </call-query>
</resource>
<resource method="GET" path="casecountry">
    <call-query href="CasebyCountryOfInfection">
        <with-param name="CountryOfInfection" query-param="CountryOfInfection"/>
    </call-query>
</resource>
<resource method="GET" path="caseinfectionacute">
    <call-query href="CasebyInfectionTypeAcute"/>
</resource>
<resource method="GET" path="caseinfectionchronic">
    <call-query href="CasebyInfectionTypeChronic"/>
</resource>
<resource method="GET" path="StatisticDate">
    <description>        Case By Statistic Date
</description>
    <call-query href="CasebyByStatic">
        <with-param name="StatisticDateFrom" query-param="StatisticDateFrom"/>
        <with-param name="StatisticDateTo" query-param="StatisticDateTo"/>
    </call-query>
</resource>
</data>

```

4.2.6 An Example of a WSDL file for one WS

In this section we provide an example of a WSDL file which is a description for DONSFed Oracle web service. This service is part of our DONSFed.

```
<wsdl2:description xmlns:wsdl2="http://www.w3.org/ns/wsdl"
xmlns:ns1="http://ws.wso2.org/dataservice"
xmlns:wsaw="http://www.w3.org/2006/05/addressing/wsdl"
xmlns:wrpc="http://www.w3.org/ns/wsdl/rpc" xmlns:wsoap="http://www.w3.org/ns/wsdl/soap"
xmlns:tns="http://www.kfupm.edu.sa/dons" xmlns:ns0="http://www.kfupm.edu.sa/dons"
xmlns:wsdlix="http://www.w3.org/ns/wsdl-extensions"
xmlns:xs="http://www.w3.org/2001/XMLSchema" xmlns:whhttp="http://www.w3.org/ns/wsdl/http"
targetNamespace="http://www.kfupm.edu.sa/dons">
<wsdl2:documentation>
Getting the data through web services from web-based database driven system using Oracle
</wsdl2:documentation>
<wsdl2:types>
<xs:schema attributeFormDefault="unqualified" elementFormDefault="qualified"
targetNamespace="http://www.kfupm.edu.sa/dons">
<xs:element name="getCaseByInfectionTypeAcute">
<xs:complexType>
<xs:sequence/>
</xs:complexType>
</xs:element>
<xs:element name="CaseInfectionTypeAcute" type="ns0:CaseInfectionTypeAcute"/>
<xs:complexType name="CaseInfectionTypeAcute">
<xs:sequence>
<xs:element ref="ns0:CaseInfectionTypeAcute"/>
</xs:sequence>
</xs:complexType>
<xs:element name="getCaseByDiagnosis">
<xs:complexType>
<xs:sequence>
<xs:element name="Diagnosis" nillable="true" type="xs:string"/>
</xs:sequence>
</xs:complexType>
</xs:element>
<xs:element name="CaseDiagnosis" type="ns0:CaseDiagnosis"/>
<xs:complexType name="CaseDiagnosis">
<xs:sequence>
<xs:element ref="ns0:CaseDiagnosis"/>
</xs:sequence>
</xs:complexType>
<xs:element name="getAllDiseases">
<xs:complexType>
<xs:sequence/>
</xs:complexType>
</xs:element>
<xs:element name="AllDiseases" type="ns0:AllDiseases"/>
<xs:complexType name="AllDiseases">
<xs:sequence>
<xs:element ref="ns0:AllDiseases"/>
</xs:sequence>
</xs:complexType>
<xs:element name="getCaseByStatisticDate">
<xs:complexType>
<xs:sequence>
<xs:element name="StatisticDateFrom" nillable="true" type="xs:date"/>
<xs:element name="StatisticDateTo" nillable="true" type="xs:date"/>
</xs:sequence>
</xs:complexType>

```

```

</xs:element>
<xs:element name="CasebyByStatic" type="ns0:CasebyByStatic"/>
<xs:complexType name="CasebyByStatic">
<xs:sequence>
<xs:element ref="ns0:CasebyByStatic"/>
</xs:sequence>
</xs:complexType>
<xs:element name="getCaseByInfectionTypeChronic">
<xs:complexType>
<xs:sequence/>
</xs:complexType>
</xs:element>
<xs:element name="CaseInfectionTypeChronic" type="ns0:CaseInfectionTypeChronic"/>
<xs:complexType name="CaseInfectionTypeChronic">
<xs:sequence>
<xs:element ref="ns0:CaseInfectionTypeChronic"/>
</xs:sequence>
</xs:complexType>
<xs:element name="getCaseByCountryOfInfection">
<xs:complexType>
<xs:sequence>
<xs:element name="CountryOfInfection" nillable="true" type="xs:string"/>
</xs:sequence>
</xs:complexType>
</xs:element>
<xs:element name="CaseCountryOfInfection" type="ns0:CaseCountryOfInfection"/>
<xs:complexType name="CaseCountryOfInfection">
<xs:sequence>
<xs:element ref="ns0:CaseCountryOfInfection"/>
</xs:sequence>
</xs:complexType>
<xs:element name="getDiseaseByName">
<xs:complexType>
<xs:sequence>
<xs:element name="name" nillable="true" type="xs:string"/>
</xs:sequence>
</xs:complexType>
</xs:element>
<xs:element name="Diseasename" type="ns0:Diseasename"/>
<xs:complexType name="Diseasename">
<xs:sequence>
<xs:element ref="ns0:Diseasename"/>
</xs:sequence>
</xs:complexType>
<xs:element name="getDiseaseByID">
<xs:complexType>
<xs:sequence>
<xs:element name="id" nillable="true" type="xs:int"/>
</xs:sequence>
</xs:complexType>
</xs:element>
<xs:element name="diseaseID" type="ns0:diseaseID"/>
<xs:complexType name="diseaseID">
<xs:sequence>
<xs:element ref="ns0:diseaseID"/>
</xs:sequence>
</xs:complexType>
<xs:element name="_getcasecountry">
<xs:complexType>
<xs:sequence>
<xs:element name="CountryOfInfection" nillable="true" type="xs:string"/>
</xs:sequence>
</xs:complexType>
</xs:element>
<xs:element name="_getcaseinfectionacute">
<xs:complexType>
<xs:sequence/>
</xs:complexType>
</xs:element>
<xs:element name="_getstatisticdate">
<xs:complexType>

```

```

<xs:sequence>
<xs:element name="StatisticDateFrom" nillable="true" type="xs:date"/>
<xs:element name="StatisticDateTo" nillable="true" type="xs:date"/>
</xs:sequence>
</xs:complexType>
</xs:element>
<xs:element name="_getdiseasename">
<xs:complexType>
<xs:sequence>
<xs:element name="name" nillable="true" type="xs:string"/>
</xs:sequence>
</xs:complexType>
</xs:element>
<xs:element name="_getcaseinfectionchronic">
<xs:complexType>
<xs:sequence/>
</xs:complexType>
</xs:element>
<xs:element name="_getdiseaseall">
<xs:complexType>
<xs:sequence/>
</xs:complexType>
</xs:element>
<xs:element name="_getdiseaseid">
<xs:complexType>
<xs:sequence>
<xs:element name="id" nillable="true" type="xs:int"/>
</xs:sequence>
</xs:complexType>
</xs:element>
<xs:element name="_getcasediagnosis">
<xs:complexType>
<xs:sequence>
<xs:element name="Diagnosis" nillable="true" type="xs:string"/>
</xs:sequence>
</xs:complexType>
</xs:element>
</xs:schema>
<xs:schema attributeFormDefault="unqualified" elementFormDefault="qualified"
targetNamespace="http://ws.wso2.org/dataservice">
<xs:element name="DataServiceFault">
<xs:complexType>
<xs:sequence>
<xs:element name="current_params" type="xs:string"/>
<xs:element name="current_request_name" type="xs:string"/>
<xs:element name="nested_exception" type="xs:string"/>
<xs:element name="source_data_service">
<xs:complexType>
<xs:sequence>
<xs:element name="location" type="xs:string"/>
<xs:element name="default_namespace" type="xs:string"/>
<xs:element name="description" type="xs:string"/>
<xs:element name="data_service_name" type="xs:string"/>
</xs:sequence>
</xs:complexType>
</xs:element>
<xs:element name="ds_code" type="xs:string"/>
</xs:sequence>
</xs:complexType>
</xs:element>
<xs:element name="REQUEST_STATUS" type="xs:string"/>
<xs:element name="DATA_SERVICE_RESPONSE">
<xs:complexType>
<xs:sequence>
<xs:any minOccurs="0"/>
</xs:sequence>
</xs:complexType>
</xs:element>
</xs:schema>
</wsdl2:types>
<wsdl2:interface name="ServiceInterface">

```

```

<wsdl2:fault name="DataServiceFault" element="ns1:DataServiceFault"/>
<wsdl2:operation name="getCaseByInfectionTypeChronic"
style="http://www.w3.org/ns/wsd1/style/iri http://www.w3.org/ns/wsd1/style/multipart"
pattern="http://www.w3.org/ns/wsd1/in-out">
<wsdl2:documentation>get Case By Infection Type Chronic</wsdl2:documentation>
<wsdl2:input element="ns0:getCaseByInfectionTypeChronic"
wsaw:Action="urn:getCaseByInfectionTypeChronic"/>
<wsdl2:output element="ns0:CaseInfectionTypeChronic"
wsaw:Action="urn:getCaseByInfectionTypeChronicResponse"/>
<wsdl2:outfault ref="tns:DataServiceFault"
wsaw:Action="urn:getCaseByInfectionTypeChronicDataServiceFault"/>
</wsdl2:operation>
<wsdl2:operation name="_getcaseinfectionacute" style="http://www.w3.org/ns/wsd1/style/iri
http://www.w3.org/ns/wsd1/style/multipart" pattern="http://www.w3.org/ns/wsd1/in-out">
<wsdl2:documentation/>
<wsdl2:input element="ns0:_getcaseinfectionacute"
wsaw:Action="urn:_getcaseinfectionacute"/>
<wsdl2:output element="ns0:CaseInfectionTypeAcute"
wsaw:Action="urn:_getcaseinfectionacuteResponse"/>
<wsdl2:outfault ref="tns:DataServiceFault"
wsaw:Action="urn:_getcaseinfectionacuteDataServiceFault"/>
</wsdl2:operation>
<wsdl2:operation name="getAllDiseases" style="http://www.w3.org/ns/wsd1/style/iri
http://www.w3.org/ns/wsd1/style/multipart" pattern="http://www.w3.org/ns/wsd1/in-out">
<wsdl2:documentation>get All Diseases</wsdl2:documentation>
<wsdl2:input element="ns0:getAllDiseases" wsaw:Action="urn:getAllDiseases"/>
<wsdl2:output element="ns0:AllDiseases" wsaw:Action="urn:getAllDiseasesResponse"/>
<wsdl2:outfault ref="tns:DataServiceFault"
wsaw:Action="urn:getAllDiseasesDataServiceFault"/>
</wsdl2:operation>
<wsdl2:operation name="getCaseByStatisticDate"
style="http://www.w3.org/ns/wsd1/style/multipart" pattern="http://www.w3.org/ns/wsd1/in-
out">
<wsdl2:documentation>get Case By Statistic Date</wsdl2:documentation>
<wsdl2:input element="ns0:getCaseByStatisticDate"
wsaw:Action="urn:getCaseByStatisticDate"/>
<wsdl2:output element="ns0:CasebyByStatic"
wsaw:Action="urn:getCaseByStatisticDateResponse"/>
<wsdl2:outfault ref="tns:DataServiceFault"
wsaw:Action="urn:getCaseByStatisticDateDataServiceFault"/>
</wsdl2:operation>
<wsdl2:operation name="_getcaseinfectionchronic"
style="http://www.w3.org/ns/wsd1/style/iri http://www.w3.org/ns/wsd1/style/multipart"
pattern="http://www.w3.org/ns/wsd1/in-out">
<wsdl2:documentation/>
<wsdl2:input element="ns0:_getcaseinfectionchronic"
wsaw:Action="urn:_getcaseinfectionchronic"/>
<wsdl2:output element="ns0:CaseInfectionTypeChronic"
wsaw:Action="urn:_getcaseinfectionchronicResponse"/>
<wsdl2:outfault ref="tns:DataServiceFault"
wsaw:Action="urn:_getcaseinfectionchronicDataServiceFault"/>
</wsdl2:operation>
<wsdl2:operation name="getDiseaseByName"
style="http://www.w3.org/ns/wsd1/style/multipart" pattern="http://www.w3.org/ns/wsd1/in-
out">
<wsdl2:documentation/>
<wsdl2:input element="ns0:getDiseaseByName" wsaw:Action="urn:getDiseaseByName"/>
<wsdl2:output element="ns0:Diseasename" wsaw:Action="urn:getDiseaseByNameResponse"/>
<wsdl2:outfault ref="tns:DataServiceFault"
wsaw:Action="urn:getDiseaseByNameDataServiceFault"/>
</wsdl2:operation>
<wsdl2:operation name="_getdiseaseall" style="http://www.w3.org/ns/wsd1/style/iri
http://www.w3.org/ns/wsd1/style/multipart" pattern="http://www.w3.org/ns/wsd1/in-out">
<wsdl2:documentation/>
<wsdl2:input element="ns0:_getdiseaseall" wsaw:Action="urn:_getdiseaseall"/>
<wsdl2:output element="ns0:AllDiseases" wsaw:Action="urn:_getdiseaseallResponse"/>
<wsdl2:outfault ref="tns:DataServiceFault"
wsaw:Action="urn:_getdiseaseallDataServiceFault"/>
</wsdl2:operation>
<wsdl2:operation name="_getcasecountry" style="http://www.w3.org/ns/wsd1/style/multipart"
pattern="http://www.w3.org/ns/wsd1/in-out">

```

```

<wsdl2:documentation/>
<wsdl2:input element="ns0:_getcasecountry" wsaw:Action="urn:_getcasecountry"/>
<wsdl2:output element="ns0:CaseCountryOfInfection"
wsaw:Action="urn:_getcasecountryResponse"/>
<wsdl2:outfault ref="tns:DataServiceFault"
wsaw:Action="urn:_getcasecountryDataServiceFault"/>
</wsdl2:operation>
<wsdl2:operation name="_getstatisticdate"
style="http://www.w3.org/ns/wsdl/style/multipart" pattern="http://www.w3.org/ns/wsdl/in-
out">
<wsdl2:documentation>Case By Statistic Date</wsdl2:documentation>
<wsdl2:input element="ns0:_getstatisticdate" wsaw:Action="urn:_getstatisticdate"/>
<wsdl2:output element="ns0:CasebyByStatic" wsaw:Action="urn:_getstatisticdateResponse"/>
<wsdl2:outfault ref="tns:DataServiceFault"
wsaw:Action="urn:_getstatisticdateDataServiceFault"/>
</wsdl2:operation>
<wsdl2:operation name="_getcasediagnosis"
style="http://www.w3.org/ns/wsdl/style/multipart" pattern="http://www.w3.org/ns/wsdl/in-
out">
<wsdl2:documentation/>
<wsdl2:input element="ns0:_getcasediagnosis" wsaw:Action="urn:_getcasediagnosis"/>
<wsdl2:output element="ns0:CaseDiagnosis" wsaw:Action="urn:_getcasediagnosisResponse"/>
<wsdl2:outfault ref="tns:DataServiceFault"
wsaw:Action="urn:_getcasediagnosisDataServiceFault"/>
</wsdl2:operation>
<wsdl2:operation name="getCaseByDiagnosis"
style="http://www.w3.org/ns/wsdl/style/multipart" pattern="http://www.w3.org/ns/wsdl/in-
out">
<wsdl2:documentation>get Case By Diagnosis</wsdl2:documentation>
<wsdl2:input element="ns0:getCaseByDiagnosis" wsaw:Action="urn:getCaseByDiagnosis"/>
<wsdl2:output element="ns0:CaseDiagnosis" wsaw:Action="urn:getCaseByDiagnosisResponse"/>
<wsdl2:outfault ref="tns:DataServiceFault"
wsaw:Action="urn:getCaseByDiagnosisDataServiceFault"/>
</wsdl2:operation>
<wsdl2:operation name="getDiseaseByID" style="http://www.w3.org/ns/wsdl/style/multipart"
pattern="http://www.w3.org/ns/wsdl/in-out">
<wsdl2:documentation>get Disease By ID</wsdl2:documentation>
<wsdl2:input element="ns0:getDiseaseByID" wsaw:Action="urn:getDiseaseByID"/>
<wsdl2:output element="ns0:diseaseID" wsaw:Action="urn:getDiseaseByIDResponse"/>
<wsdl2:outfault ref="tns:DataServiceFault"
wsaw:Action="urn:getDiseaseByIDDDataServiceFault"/>
</wsdl2:operation>
<wsdl2:operation name="_getdiseasename" style="http://www.w3.org/ns/wsdl/style/multipart"
pattern="http://www.w3.org/ns/wsdl/in-out">
<wsdl2:documentation/>
<wsdl2:input element="ns0:_getdiseasename" wsaw:Action="urn:_getdiseasename"/>
<wsdl2:output element="ns0:Diseasename" wsaw:Action="urn:_getdiseasenameResponse"/>
<wsdl2:outfault ref="tns:DataServiceFault"
wsaw:Action="urn:_getdiseasenameDataServiceFault"/>
</wsdl2:operation>
<wsdl2:operation name="getCaseByCountryOfInfection"
style="http://www.w3.org/ns/wsdl/style/multipart" pattern="http://www.w3.org/ns/wsdl/in-
out">
<wsdl2:documentation>get Case By Country Of Infection</wsdl2:documentation>
<wsdl2:input element="ns0:getCaseByCountryOfInfection"
wsaw:Action="urn:getCaseByCountryOfInfection"/>
<wsdl2:output element="ns0:CaseCountryOfInfection"
wsaw:Action="urn:getCaseByCountryOfInfectionResponse"/>
<wsdl2:outfault ref="tns:DataServiceFault"
wsaw:Action="urn:getCaseByCountryOfInfectionDataServiceFault"/>
</wsdl2:operation>
<wsdl2:operation name="getCaseByInfectionTypeAcute"
style="http://www.w3.org/ns/wsdl/style/iri http://www.w3.org/ns/wsdl/style/multipart"
pattern="http://www.w3.org/ns/wsdl/in-out">
<wsdl2:documentation>get Case By Infection Type Acute</wsdl2:documentation>
<wsdl2:input element="ns0:getCaseByInfectionTypeAcute"
wsaw:Action="urn:getCaseByInfectionTypeAcute"/>
<wsdl2:output element="ns0:CaseInfectionTypeAcute"
wsaw:Action="urn:getCaseByInfectionTypeAcuteResponse"/>
<wsdl2:outfault ref="tns:DataServiceFault"
wsaw:Action="urn:getCaseByInfectionTypeAcuteDataServiceFault"/>

```

```

</wsdl2:operation>
<wsdl2:operation name="_getdiseaseid" style="http://www.w3.org/ns/wsdl/style/multipart"
pattern="http://www.w3.org/ns/wsdl/in-out">
<wsdl2:documentation/>
<wsdl2:input element="ns0:_getdiseaseid" wsaw:Action="urn:_getdiseaseid"/>
<wsdl2:output element="ns0:diseaseID" wsaw:Action="urn:_getdiseaseidResponse"/>
<wsdl2:outfault ref="tns:DataServiceFault"
wsaw:Action="urn:_getdiseaseidDataServiceFault"/>
</wsdl2:operation>
</wsdl2:interface>
<wsdl2:binding name="DonsFedOracleHttpBinding" interface="tns:ServiceInterface"
type="http://www.w3.org/ns/wsdl/http">
<wsdl2:operation ref="tns:getCaseByInfectionTypeChronic" whttp:method="POST"
whttp:location="getCaseByInfectionTypeChronic">
<wsdl2:output/>
<wsdl2:output/>
<wsdl2:outfault ref="tns:DataServiceFault"/>
</wsdl2:operation>
<wsdl2:operation ref="tns:_getcaseinfectionacute" whttp:method="GET"
whttp:location="caseinfectionacute">
<wsdl2:output/>
<wsdl2:output/>
<wsdl2:outfault ref="tns:DataServiceFault"/>
</wsdl2:operation>
<wsdl2:operation ref="tns:getAllDiseases" whttp:method="POST"
whttp:location="getAllDiseases">
<wsdl2:output/>
<wsdl2:output/>
<wsdl2:outfault ref="tns:DataServiceFault"/>
</wsdl2:operation>
<wsdl2:operation ref="tns:getCaseByStatisticDate" whttp:method="POST"
whttp:location="getCaseByStatisticDate">
<wsdl2:output/>
<wsdl2:output/>
<wsdl2:outfault ref="tns:DataServiceFault"/>
</wsdl2:operation>
<wsdl2:operation ref="tns:_getcaseinfectionchronic" whttp:method="GET"
whttp:location="caseinfectionchronic">
<wsdl2:output/>
<wsdl2:output/>
<wsdl2:outfault ref="tns:DataServiceFault"/>
</wsdl2:operation>
<wsdl2:operation ref="tns:getDiseaseByName" whttp:method="POST"
whttp:location="getDiseaseByName">
<wsdl2:output/>
<wsdl2:output/>
<wsdl2:outfault ref="tns:DataServiceFault"/>
</wsdl2:operation>
<wsdl2:operation ref="tns:_getdiseaseall" whttp:method="GET" whttp:location="diseaseall">
<wsdl2:output/>
<wsdl2:output/>
<wsdl2:outfault ref="tns:DataServiceFault"/>
</wsdl2:operation>
<wsdl2:operation ref="tns:_getcasecountry" whttp:method="GET"
whttp:location="casecountry">
<wsdl2:output/>
<wsdl2:output/>
<wsdl2:outfault ref="tns:DataServiceFault"/>
</wsdl2:operation>
<wsdl2:operation ref="tns:_getstatisticdate" whttp:method="GET"
whttp:location="StatisticDate">
<wsdl2:output/>
<wsdl2:output/>
<wsdl2:outfault ref="tns:DataServiceFault"/>
</wsdl2:operation>
<wsdl2:operation ref="tns:_getcasediagnosis" whttp:method="GET"
whttp:location="casediagnosis">
<wsdl2:output/>
<wsdl2:output/>
<wsdl2:outfault ref="tns:DataServiceFault"/>
</wsdl2:operation>

```

```

<wsdl2:operation ref="tns:getCaseByDiagnosis" http:method="POST"
http:location="getCaseByDiagnosis">
<wsdl2:output/>
<wsdl2:output/>
<wsdl2:outfault ref="tns:DataServiceFault"/>
</wsdl2:operation>
<wsdl2:operation ref="tns:getDiseaseByID" http:method="POST"
http:location="getDiseaseByID">
<wsdl2:output/>
<wsdl2:output/>
<wsdl2:outfault ref="tns:DataServiceFault"/>
</wsdl2:operation>
<wsdl2:operation ref="tns:_getdiseasename" http:method="GET"
http:location="diseasename">
<wsdl2:output/>
<wsdl2:output/>
<wsdl2:outfault ref="tns:DataServiceFault"/>
</wsdl2:operation>
<wsdl2:operation ref="tns:getCaseByCountryOfInfection" http:method="POST"
http:location="getCaseByCountryOfInfection">
<wsdl2:output/>
<wsdl2:output/>
<wsdl2:outfault ref="tns:DataServiceFault"/>
</wsdl2:operation>
<wsdl2:operation ref="tns:getCaseByInfectionTypeAcute" http:method="POST"
http:location="getCaseByInfectionTypeAcute">
<wsdl2:output/>
<wsdl2:output/>
<wsdl2:outfault ref="tns:DataServiceFault"/>
</wsdl2:operation>
<wsdl2:operation ref="tns:_getdiseaseid" http:method="GET" http:location="diseaseid">
<wsdl2:output/>
<wsdl2:output/>
<wsdl2:outfault ref="tns:DataServiceFault"/>
</wsdl2:operation>
</wsdl2:binding>
<wsdl2:binding name="DonsFedOracleSOAP12Binding" interface="tns:ServiceInterface"
type="http://www.w3.org/ns/wsdl/soap" soap:version="1.2"
soap:protocol="http://www.w3.org/2003/05/soap/bindings/HTTP">
<wsdl2:operation ref="tns:getCaseByInfectionTypeChronic"
soap:action="urn:getCaseByInfectionTypeChronic"
http:location="getCaseByInfectionTypeChronic">
<wsdl2:output/>
<wsdl2:output/>
<wsdl2:outfault ref="tns:DataServiceFault"/>
</wsdl2:operation>
<wsdl2:operation ref="tns:_getcaseinfectionacute"
soap:action="urn:_getcaseinfectionacute" http:location="caseinfectionacute">
<wsdl2:output/>
<wsdl2:output/>
<wsdl2:outfault ref="tns:DataServiceFault"/>
</wsdl2:operation>
<wsdl2:operation ref="tns:getAllDiseases" soap:action="urn:getAllDiseases"
http:location="getAllDiseases">
<wsdl2:output/>
<wsdl2:output/>
<wsdl2:outfault ref="tns:DataServiceFault"/>
</wsdl2:operation>
<wsdl2:operation ref="tns:getCaseByStatisticDate"
soap:action="urn:getCaseByStatisticDate" http:location="getCaseByStatisticDate">
<wsdl2:output/>
<wsdl2:output/>
<wsdl2:outfault ref="tns:DataServiceFault"/>
</wsdl2:operation>
<wsdl2:operation ref="tns:_getcaseinfectionchronic"
soap:action="urn:_getcaseinfectionchronic" http:location="caseinfectionchronic">
<wsdl2:output/>
<wsdl2:output/>
<wsdl2:outfault ref="tns:DataServiceFault"/>
</wsdl2:operation>

```

```

<wsdl2:operation ref="tns:getDiseaseByName" wsoap:action="urn:getDiseaseByName"
whhttp:location="getDiseaseByName">
<wsdl2:output/>
<wsdl2:output/>
<wsdl2:outfault ref="tns:DataServiceFault"/>
</wsdl2:operation>
<wsdl2:operation ref="tns:_getdiseaseall" wsoap:action="urn:_getdiseaseall"
whhttp:location="diseaseall">
<wsdl2:output/>
<wsdl2:output/>
<wsdl2:outfault ref="tns:DataServiceFault"/>
</wsdl2:operation>
<wsdl2:operation ref="tns:_getcasecountry" wsoap:action="urn:_getcasecountry"
whhttp:location="casecountry">
<wsdl2:output/>
<wsdl2:output/>
<wsdl2:outfault ref="tns:DataServiceFault"/>
</wsdl2:operation>
<wsdl2:operation ref="tns:_getstatisticdate" wsoap:action="urn:_getstatisticdate"
whhttp:location="StatisticDate">
<wsdl2:output/>
<wsdl2:output/>
<wsdl2:outfault ref="tns:DataServiceFault"/>
</wsdl2:operation>
<wsdl2:operation ref="tns:_getcasediagnosis" wsoap:action="urn:_getcasediagnosis"
whhttp:location="casediagnosis">
<wsdl2:output/>
<wsdl2:output/>
<wsdl2:outfault ref="tns:DataServiceFault"/>
</wsdl2:operation>
<wsdl2:operation ref="tns:getCaseByDiagnosis" wsoap:action="urn:getCaseByDiagnosis"
whhttp:location="getCaseByDiagnosis">
<wsdl2:output/>
<wsdl2:output/>
<wsdl2:outfault ref="tns:DataServiceFault"/>
</wsdl2:operation>
<wsdl2:operation ref="tns:getDiseaseByID" wsoap:action="urn:getDiseaseByID"
whhttp:location="getDiseaseByID">
<wsdl2:output/>
<wsdl2:output/>
<wsdl2:outfault ref="tns:DataServiceFault"/>
</wsdl2:operation>
<wsdl2:operation ref="tns:_getdiseasename" wsoap:action="urn:_getdiseasename"
whhttp:location="diseasename">
<wsdl2:output/>
<wsdl2:output/>
<wsdl2:outfault ref="tns:DataServiceFault"/>
</wsdl2:operation>
<wsdl2:operation ref="tns:getCaseByCountryOfInfection"
wsoap:action="urn:getCaseByCountryOfInfection"
whhttp:location="getCaseByCountryOfInfection">
<wsdl2:output/>
<wsdl2:output/>
<wsdl2:outfault ref="tns:DataServiceFault"/>
</wsdl2:operation>
<wsdl2:operation ref="tns:getCaseByInfectionTypeAcute"
wsoap:action="urn:getCaseByInfectionTypeAcute"
whhttp:location="getCaseByInfectionTypeAcute">
<wsdl2:output/>
<wsdl2:output/>
<wsdl2:outfault ref="tns:DataServiceFault"/>
</wsdl2:operation>
<wsdl2:operation ref="tns:_getdiseaseid" wsoap:action="urn:_getdiseaseid"
whhttp:location="diseaseid">
<wsdl2:output/>
<wsdl2:output/>
<wsdl2:outfault ref="tns:DataServiceFault"/>
</wsdl2:operation>
</wsdl2:binding>

```

```

<wsdl2:binding name="DonsFedOracleSOAP11Binding" interface="tns:ServiceInterface"
type="http://www.w3.org/ns/wsdl/soap" wsdl:version="1.1"
wsdl:protocol="http://www.w3.org/2003/05/soap/bindings/HTTP">
<wsdl2:operation ref="tns:getCaseByInfectionTypeChronic"
wsdl:action="urn:getCaseByInfectionTypeChronic"
wsdl:location="getCaseByInfectionTypeChronic">
<wsdl2:output/>
<wsdl2:output/>
<wsdl2:outfault ref="tns:DataServiceFault"/>
</wsdl2:operation>
<wsdl2:operation ref="tns:_getcaseinfectionacute"
wsdl:action="urn:_getcaseinfectionacute" wsdl:location="caseinfectionacute">
<wsdl2:output/>
<wsdl2:output/>
<wsdl2:outfault ref="tns:DataServiceFault"/>
</wsdl2:operation>
<wsdl2:operation ref="tns:getAllDiseases" wsdl:action="urn:getAllDiseases"
wsdl:location="getAllDiseases">
<wsdl2:output/>
<wsdl2:output/>
<wsdl2:outfault ref="tns:DataServiceFault"/>
</wsdl2:operation>
<wsdl2:operation ref="tns:getCaseByStatisticDate"
wsdl:action="urn:getCaseByStatisticDate" wsdl:location="getCaseByStatisticDate">
<wsdl2:output/>
<wsdl2:output/>
<wsdl2:outfault ref="tns:DataServiceFault"/>
</wsdl2:operation>
<wsdl2:operation ref="tns:_getcaseinfectionchronic"
wsdl:action="urn:_getcaseinfectionchronic" wsdl:location="caseinfectionchronic">
<wsdl2:output/>
<wsdl2:output/>
<wsdl2:outfault ref="tns:DataServiceFault"/>
</wsdl2:operation>
<wsdl2:operation ref="tns:getDiseaseByName" wsdl:action="urn:getDiseaseByName"
wsdl:location="getDiseaseByName">
<wsdl2:output/>
<wsdl2:output/>
<wsdl2:outfault ref="tns:DataServiceFault"/>
</wsdl2:operation>
<wsdl2:operation ref="tns:_getdiseaseall" wsdl:action="urn:_getdiseaseall"
wsdl:location="diseaseall">
<wsdl2:output/>
<wsdl2:output/>
<wsdl2:outfault ref="tns:DataServiceFault"/>
</wsdl2:operation>
<wsdl2:operation ref="tns:_getcasecountry" wsdl:action="urn:_getcasecountry"
wsdl:location="casecountry">
<wsdl2:output/>
<wsdl2:output/>
<wsdl2:outfault ref="tns:DataServiceFault"/>
</wsdl2:operation>
<wsdl2:operation ref="tns:_getstatisticdate" wsdl:action="urn:_getstatisticdate"
wsdl:location="StatisticDate">
<wsdl2:output/>
<wsdl2:output/>
<wsdl2:outfault ref="tns:DataServiceFault"/>
</wsdl2:operation>
<wsdl2:operation ref="tns:_getcasediagnosis" wsdl:action="urn:_getcasediagnosis"
wsdl:location="casediagnosis">
<wsdl2:output/>
<wsdl2:output/>
<wsdl2:outfault ref="tns:DataServiceFault"/>
</wsdl2:operation>
<wsdl2:operation ref="tns:getCaseByDiagnosis" wsdl:action="urn:getCaseByDiagnosis"
wsdl:location="getCaseByDiagnosis">
<wsdl2:output/>
<wsdl2:output/>
<wsdl2:outfault ref="tns:DataServiceFault"/>
</wsdl2:operation>

```

```

<wsdl2:operation ref="tns:getDiseaseByID" wsoap:action="urn:getDiseaseByID"
whhttp:location="getDiseaseByID">
<wsdl2:output/>
<wsdl2:output/>
<wsdl2:outfault ref="tns:DataServiceFault"/>
</wsdl2:operation>
<wsdl2:operation ref="tns:_getdiseasename" wsoap:action="urn:_getdiseasename"
whhttp:location="diseasename">
<wsdl2:output/>
<wsdl2:output/>
<wsdl2:outfault ref="tns:DataServiceFault"/>
</wsdl2:operation>
<wsdl2:operation ref="tns:getCaseByCountryOfInfection"
wsoap:action="urn:getCaseByCountryOfInfection"
whhttp:location="getCaseByCountryOfInfection">
<wsdl2:output/>
<wsdl2:output/>
<wsdl2:outfault ref="tns:DataServiceFault"/>
</wsdl2:operation>
<wsdl2:operation ref="tns:getCaseByInfectionTypeAcute"
wsoap:action="urn:getCaseByInfectionTypeAcute"
whhttp:location="getCaseByInfectionTypeAcute">
<wsdl2:output/>
<wsdl2:output/>
<wsdl2:outfault ref="tns:DataServiceFault"/>
</wsdl2:operation>
<wsdl2:operation ref="tns:_getdiseaseid" wsoap:action="urn:_getdiseaseid"
whhttp:location="diseaseid">
<wsdl2:output/>
<wsdl2:output/>
<wsdl2:outfault ref="tns:DataServiceFault"/>
</wsdl2:operation>
</wsdl2:binding>
<wsdl2:service name="DonsFedOracle" interface="tns:ServiceInterface">
<wsdl2:endpoint name="HTTPEndpoint" binding="tns:DonsFedOracleHttpBinding"
address="http://10.80.9.198:9763/services/DonsFedOracle.HTTPEndpoint/">
<wsdl2:endpoint name="SecureSOAP12Endpoint" binding="tns:DonsFedOracleSOAP12Binding"
address="https://10.80.9.198:9443/services/DonsFedOracle.SecureSOAP12Endpoint/">
<wsdl2:endpoint name="SOAP11Endpoint" binding="tns:DonsFedOracleSOAP11Binding"
address="http://10.80.9.198:9763/services/DonsFedOracle.SOAP11Endpoint/">
<wsdl2:endpoint name="SecureHTTPEndpoint" binding="tns:DonsFedOracleHttpBinding"
address="https://10.80.9.198:9443/services/DonsFedOracle.SecureHTTPEndpoint/">
<wsdl2:endpoint name="SecureSOAP11Endpoint" binding="tns:DonsFedOracleSOAP11Binding"
address="https://10.80.9.198:9443/services/DonsFedOracle.SecureSOAP11Endpoint/">
<wsdl2:endpoint name="SOAP12Endpoint" binding="tns:DonsFedOracleSOAP12Binding"
address="http://10.80.9.198:9763/services/DonsFedOracle.SOAP12Endpoint/">
</wsdl2:service>
</wsdl2:description>

```

CHAPTER 5

CONCLUSION AND FUTURE WORK

In this thesis, we have shown that for the integration process, a federated database approach using web services is suitable for many reasons. Here are some of benefits to use such approach: the components of federated system can be added and removed without degrading the overall federating system. Also, the ability to access, share, and retrieve data from each component system is supported. By using the web services, the structure of the constituent databases can be abstracted using XML or any other semantic technique. Furthermore, flexibility of creating a dynamic federation of DBs enables maintaining the autonomous component systems and handles the heterogeneity of the distributed databases. Additionally, it eliminates the need for local to global schema translation and supports both compliant and non-compliant databases.

We presented a federated databases framework for outbreak information and notification systems. This framework was implemented by using web services as an integration method. We provided a proof-of-concept implementation to the proposed framework. We used three different autonomous and distributed databases. The first one was obtained from the KSA DONS system which is an oracle cloud-based database. The second one was acquired from the CASE system which is the open source MySQL database. The third database is based on SQL Server. These databases which are located at different locations, have different schemas, and different semantics, and are therefore suitable for

testing our federation framework. Finally, the experimental results were presented and analyzed in this thesis.

As a future work, to make the DONSFed framework works for every new DONS system to be part of the federation, we have to develop annotations. Once these annotations exist, the federated and constituent systems only must agree on the developed ontology to decrease an ambiguity of semantic. The semantics is controlled by the federation and this can be done by annotating terms using WSDL annotated from developed ontology. These annotations offer meaning to the functionality of each operation, inputs and outputs of a Web service.

References

- [1] Z. Lacroix, B. Ludäscher, and R. Stevens, *Part 10 Basic Bioinformatics Technologies Integrating Biological Databases*, vol. 3. 2007, pp. 1525–1571.
- [2] M. Muller and S. Richardson, “Early diagnosis of SARS: lessons from the Toronto SARS outbreak,” *Eur. J. Clin. Microbiol. Infect. Dis.*, vol. 25, no. 4, pp. 230–237, 2006.
- [3] C. Leung, M. Ho, and A. Kiss, “Homelessness and the response to emerging infectious disease outbreaks: lessons from SARS,” *J. Urban Heal.*, vol. 85, no. 3, pp. 402–410, 2008.
- [4] N. Cohen, J. Morita, D. Plate, and R. Jones, “Control of an outbreak due to an adamantane-resistant strain of influenza A (H3N2) in a chronic care facility,” *Infection*, 2008.
- [5] G. Ackerman, *Bioterrorism and threat assessment*. 2004, p. Available at: <http://www.blixassociates.com/wp-con>.
- [6] M. Donohoe, “Internists, epidemics, outbreaks, and bioterrorist attacks,” *J. Gen. Intern. Med.*, vol. 22, no. 9, p. 1380, 2007.
- [7] *Guidelines for Outbreak Prevention , Control and Management in Acute Care and Facility Living Sites*. Alberta Health Services, 2013, pp. 1–54.
- [8] G. Cruz-Pacheco, L. Esteva, and C. Vargas, “Seasonality and outbreaks in West Nile virus infection,” *Bull. Math. Biol.*, vol. 71, no. 6, pp. 1378–93, Aug. 2009.
- [9] D. Tam, S. Lee, and S. Lee, “Impact of SARS on avian influenza preparedness in healthcare workers,” *Infection*, vol. 3, no. 5, pp. 320–325, 2007.
- [10] “WHO | Influenza at the Human-Animal Interface (HAI).” [Online]. Available: http://www.who.int/influenza/human_animal_interface/en/. [Accessed: 11-Feb-2014].
- [11] T. W. Organisation and A. Health, *The World Organisation for Animal Health (OIE) Prevention and control of animal diseases worldwide Economic analysis – Prevention versus outbreak costs Final Report Economic analysis – Prevention*

versus outbreak costs Final Report Part I. Civic Consulting • Agra CEAS Consulting, 2007.

- [12] Centers for Disease Control and Prevention (CDC)., “Summary of notifiable diseases--United States, 2010.,” *MMWR. Morb. Mortal. Wkly. Rep.*, p. 58.53, 2012.
- [13] J. Wang, Z. Miao, Y. Zhang, and B. Zhou, “Querying heterogeneous relational database using SPARQL,” *Comput. Inf. ...*, 2009.
- [14] Y. Wang, “The Research of Multi-source heterogeneous Data Integration Based on LINQ,” 2012.
- [15] V. M. Shelake, “A Novel Approach for Multi-Source Heterogeneous Database Integration,” 2013.
- [16] D. Waga, E. Makori, and K. Rabah, “Utilization of Cloud Computing in Education and Research to the Attainment of Millennium Development Goals and Vision 2030 in Kenya,” vol. 2, no. 2, pp. 193–199, 2014.
- [17] P. Kumar, “An Overview of Architectures and Techniques for Integrated Data Systems (IDS) Implementation,” 2013.
- [18] R. F. Van Der Lans, I. Business, and I. Analyst, *Data Services : The Marriage of Data Integration and Application Integration.* 2012.
- [19] Y. Wang, H. Liu, and I. Science, “Research on Data Integration Based on Cloud Computing,” vol. 6, no. 3, pp. 433–436, 2013.
- [20] S. Hellmann, J. Lehmann, and M. Br, *Integrating NLP using Linked Data.* The Semantic Web – ISWC 2013, 2013, pp. 1–16.
- [21] B. Zhou, “Data integration as a service for Applications Deployment on the SaaS Platform,” *Biomed. Eng. Informatics (BMEI), 2013 ...*, 2013.
- [22] P. D. Karp, “A Strategy for Database integration,” *J. Comput. Biol.*, vol. 2, no. 4, pp. 573–586, 1995.
- [23] Z. Wang, X. Gao, C. He, J. a. Miller, J. C. Kissinger, M. Heiges, and E. T. Kraemer, “An Evaluation of Multiple Approaches for Federating Biological Data,” *J. Inf. Technol. Res.*, vol. 2, no. 2, pp. 42–64, 2009.
- [24] S. Philippi, “Light-weight integration of molecular biological databases,” *Bioinformatics*, vol. 20, no. 1, pp. 51–57, Dec. 2003.

- [25] C. Schonbach, P. Kowalski-saunders, and V. Brusica, "Data Warehousing in molecular biology," *Brief. Bioinform.*, vol. 1.2, pp. 190–198., 2000.
- [26] A. Noaman, F. Essia, and M. Salah, "Web Services Based Integration Tool for Heterogeneous Databases," *Int. J. Res. Eng. Sci.*, vol. 1, no. 3, pp. 16–26, 2013.
- [27] M. Barhamgi, F. Cuppens, and B. Defude, "PAIRSE : A Privacy-Preserving Service-Oriented Data Integration System," *SIGMOD Rec.*, vol. 42, no. 3, 2013.
- [28] C. A. Bradley, H. Rolka, D. Walker, J. Loonsk, and others, "BioSense: implementation of a national early event detection and situational awareness system," *MMWR Morb Mortal Wkly Rep*, vol. 54, no. Suppl, pp. 11–19, 2005.
- [29] B. Cakici, K. Hebing, M. Grünewald, P. Saretok, and A. Hulth, "CASE: a framework for computer supported outbreak detection," *BMC Med. Inform. Decis. Mak.*, vol. 10, no. 1, p. 14, 2010.
- [30] J. S. Brownstein, C. C. Freifeld, B. Y. Reis, and K. D. Mandl, "Surveillance Sans Frontieres: Internet-based emerging infectious disease intelligence and the HealthMap project," *PLoS Med.*, vol. 5, no. 7, p. e151, 2008.
- [31] D. McLeod and D. Heimbigner, "A federated architecture for database systems," *Proc. May 19-22, 1980, Natl. Comput. Conf. - AFIPS '80*, p. 283, 1980.
- [32] J. A. Larson, "Federated Database Systems for Managing Distributed , Heterogeneous , and Autonomous Databases ' Amit Sheth and James Larson," vol. 22, no. 3, 1990.
- [33] E. Wheeler, D. L., Barrett, T., Benson, D. A., Bryant, S. H., Canese, K., Chetvernin, V., ... & Yaschenko, "Database resources of the National Center for Biotechnology Information.," *Nucleic Acids Res.*, vol. 37, no. Database issue, pp. D5–15, Jan. 2009.
- [34] J. Zhang, S. Haider, J. Baran, A. Cros, J. M. Guberman, J. Hsu, Y. Liang, L. Yao, and A. Kasprzyk, "Original article BioMart : a data federation framework for large collaborative projects," *Database 2011*, pp. 1–15, 2011.
- [35] A. Kasprzyk, "BioMart: driving a paradigm change in biological data management.," *Database (Oxford)*, vol. 2011, p. bar049, Jan. 2011.
- [36] D. Smedley, S. Haider, B. Ballester, R. Holland, D. London, G. Thorisson, and A. Kasprzyk, "BioMart--biological queries made easy.," *BMC Genomics*, vol. 10, p. 22, Jan. 2009.
- [37] "EuPathDB : The Eukaryotic Pathogen genome resource." [Online]. Available: <http://eupathdb.org/eupathdb/>. [Accessed: 22-Nov-2014].

- [38] C. Aurrecochea and A. Barreto, “AmoebaDB and MicrosporidiaDB: functional genomic resources for Amoebozoa and Microsporidia species,” *Nucleic acids ...*, 2011.
- [39] C. Aurrecochea, J. Brestelli, B. P. Brunk, S. Fischer, B. Gajria, X. Gao, A. Gingle, G. Grant, O. S. Harb, M. Heiges, F. Innamorato, J. Iodice, J. C. Kissinger, E. T. Kraemer, W. Li, J. a Miller, V. Nayak, C. Pennington, D. F. Pinney, D. S. Roos, C. Ross, G. Srinivasamoorthy, C. J. Stoeckert, R. Thibodeau, C. Treatman, and H. Wang, “EuPathDB: a portal to eukaryotic pathogen databases.,” *Nucleic Acids Res.*, vol. 38, no. Database issue, pp. D415–9, Jan. 2010.
- [40] C. Aurrecochea, A. Barreto, J. Brestelli, B. P. Brunk, S. Cade, R. Doherty, S. Fischer, B. Gajria, X. Gao, A. Gingle, G. Grant, O. S. Harb, M. Heiges, S. Hu, J. Iodice, J. C. Kissinger, E. T. Kraemer, W. Li, D. F. Pinney, B. Pitts, D. S. Roos, G. Srinivasamoorthy, C. J. Stoeckert, H. Wang, and S. Warrenfeltz, “EuPathDB: the eukaryotic pathogen database.,” *Nucleic Acids Res.*, vol. 41, no. Database issue, pp. D684–91, Jan. 2013.
- [41] Q. Sheng, X. Qiao, A. Vasilakos, and C. Szabo, “Web services composition: A decade’s overview,” *Inf. ...*, 2014.
- [42] H. Wang, J. Huang, Y. Qu, and J. Xie, “Web services: problems and future directions,” *Web Semant. Sci. Serv. ...*, 2004.
- [43] D. Austin, A. Barbir, C. Ferris, and S. Garg, “Web services architecture requirements,” *W3C Work. Gr. Notes*, 2004.
- [44] W. Services and C. Architecture, “Web Services Conceptual Architecture (WSCA 1.0),” *IBM Softw. Gr. 5*, 2001.
- [45] M. Mehta, S. Lee, and J. Shah, “Service-Oriented Architecture: Concepts and Implementation,” *Proc. Inf. ...*, 2006.
- [46] S. Abeysinghe, *RESTful PHP Web Services*. 2008.
- [47] L. Richardson and S. Ruby, *RESTful web services*. 2008.
- [48] S. Krishnan, B. Stearn, and K. Bhatia, “Web services-based data integration for life science computations,” ... *Cent. Tech. Rep. ...*, 2005.
- [49] “API Web services, WSDL, SDK, XML, SOAP .NET Web applications.” [Online]. Available: <http://www.webservicelist.com/>. [Accessed: 23-Nov-2014].
- [50] “ProgrammableWeb - APIs, Mashups and the Web as Platform.” [Online]. Available: <http://www.programmableweb.com/>. [Accessed: 23-Nov-2014].

- [51] A. Bouguettaya, Q. Sheng, and F. Daniel, *Advanced web services*. 2013.
- [52] Q. Duan, Y. Yan, and A. Vasilakos, “A survey on service-oriented network virtualization toward convergence of networking and cloud computing,” *Netw. Serv.*, 2012.
- [53] S. Mathew, Y. Atif, Q. Sheng, and Z. Maamar, “The Web of Things-Challenges and Enabling Technologies,” *Internet things inter-*, 2013.
- [54] “Web Services Description Language (WSDL) Version 2.0 Part 1: Core Language.” [Online]. Available: <http://www.w3.org/TR/wsdl20/>. [Accessed: 23-Nov-2014].
- [55] “SOAP Specifications.” [Online]. Available: <http://www.w3.org/TR/soap/>. [Accessed: 23-Nov-2014].
- [56] Y.-Y. Peng, S.-P. Ma, and J. Lee, “REST2SOAP: A framework to integrate SOAP services and RESTful services,” *2009 IEEE Int. Conf. Serv. Comput. Appl.*, pp. 1–4, Dec. 2009.
- [57] F. Curbera, M. Duftler, R. Khalaf, and W. Nagy, “Unraveling the web services web,” *IEEE Internet Comput.* 6.2, pp. 86–93, 2002.
- [58] C. Pautasso and F. Leymann, “RESTful Web Services vs . ‘ Big ’ Web Services : Making the Right Architectural Decision Categories and Subject Descriptors,” pp. 805–814, 2008.
- [59] F. Curbera, R. Khalaf, and N. Mukhi, “The next step in web services,” *Commun.*, 2003.
- [60] E. C. C. Meredith;S, “Web Service Definition Language (WSDL),” 2001. [Online]. Available: <http://www.w3.org/TR/wsdl/>. [Accessed: 24-Jan-2014].
- [61] R. Aggarwal, K. Verma, J. Miller, and W. Milnor, “Constraint driven web service composition in METEOR-S,” *IEEE Int. Conf. onServices Comput. 2004. (SCC 2004). Proceedings. 2004*, pp. 23–30.
- [62] “Ladon Attachments - Ladon Webservice.” [Online]. Available: http://ladonize.org/index.php/Ladon_Attachments. [Accessed: 28-Jan-2014].
- [63] D. Crockford, “The application/json Media Type for JavaScript Object Notation (JSON),” pp. 1–11, 2006.
- [64] H. Hellbruck, T. Teubler, and S. Fischer, “Name-Centric Service Architecture for Cyber-Physical Systems (Short Paper),” *2013 IEEE 6th Int. Conf. Serv. Comput. Appl.*, pp. 77–82, Dec. 2013.

- [65] G. Alonso, F. Casati, H. Kuno, and V. Machiraju, “Web Services: Concepts, architectures and applications. 2004,” *Springer*.
- [66] “National Center for Biotechnology Information.” [Online]. Available: <http://www.ncbi.nlm.nih.gov/>. [Accessed: 23-Nov-2014].
- [67] “DDBJ | DNA Data Bank of Japan.” [Online]. Available: <http://www.ddbj.nig.ac.jp/>. [Accessed: 23-Nov-2014].
- [68] “EMBL European Bioinformatics Institute.” [Online]. Available: <http://www.ebi.ac.uk/>. [Accessed: 23-Nov-2014].
- [69] Consortium_BioMoby, “Interoperability with Moby 1.0--it’s better than sharing your toothbrush!,” *Brief. Bioinform.*, vol. 9, no. 3, pp. 220–31, May 2008.
- [70] M. D. Wilkinson and M. Links, “BioMOBY: an open source biological web services proposal,” *Brief. Bioinform.*, vol. 3, no. 4, pp. 331–41, Dec. 2002.
- [71] B. W. S. Boyu Yang, Tian Xue, Jing Zhao, Chaitanya Kommidi, Jeetendra Soneja, Jian Li, Rebecca Will, Bruce Sharp, Ron Kenyon, Oswald Crasta, “Bioinformatics Web Services,” *Pap. Present. 2006 Int. Conf. Bioinformatics.*, 2006.
- [72] C. Pennington, “Building federated bioinformatics databases using Web services,” 2009.
- [73] K. Goundar, S. Singh, and X. Ye, “An investigation into concurrency control mechanisms in data service layers,” *Softw. Eng. Conf. ...*, 2007.
- [74] S. Gannouni, M. Beraka, and H. Mathkour, “DSM: a data service middleware for sharing data in peer-to-peer computing environments,” *Int. J. Innov. ...*, 2012.
- [75] “WSO2 Data Services Server .” [Online]. Available: <http://wso2.com/products/data-services-server/>. [Accessed: 29-Nov-2014].

Vitae

Name : Mustafa Mohammed Saeed Ghaleb

Nationality : Yemeni

Date of Birth : 1/5/1983

Email : alwaridi2@gmail.com

Address : Yemen, Taiz

Education and Qualifications:

- Bachelor Degree in Education (King Khaled University - Abha): - Abha Collage of Teachers – Computer (2006-2007). - GPA: Excellent (4.99 out of 5).
- Master Degree (KFUPM): - Information and Computer Science (2014). - GPA: Very Good (3.406 out of 4).

Published Papers:

- Farag Azzedin, Jaweed Yazdani, Salahadin Mohammed, Mustafa Ghaleb, “Generic Model for Disease Outbreak Notification Systems”, International Journal of Computer Science & Information Technology (IJCSIT), ISSN: 0975 - 4660, Vol 6, No 4, August 2014.

- Azzedin, Farag, Jaweed Yazdani, and Mustafa Ghaleb. "Survey of Disease Outbreak Notification Systems." *Advances in Computing, Communications and Informatics (ICACCI)*, 2013 International Conference on. IEEE, 2013.
- Farag Azzedin, Salahadin Mohammed, Jaweed Yazdani, Mustafa Ghaleb, "Designing a Disease Outbreak Notification System in Saudi Arabia", Second International Conference of Advanced Computer Science & Information Technology (ACSIT-2014), June 14-15, 2014, Zurich, Switzerland, (<http://airccse.org/cscp.html>)