



Letterkenny Institute of Technology

A thesis submitted in partial fulfilment of
the requirements for the Master of Science in Computing in
Enterprise Applications Development Letterkenny Institute of Technology

A preliminary exploration of database performance for use with 'Big Data' projects in the aviation industry

Author:
Maria Griffin

Supervisor:
Nigel McKelvey

Submitted to the Higher Education and Training Awards Council (HETAC)

June 2014

Declaration

I hereby certify that the material, which I now submit for assessment on the programmes of study leading to the award of ***Master of Science in Computing in Enterprise Application Development***, is entirely my own work and has not been taken from the work of others except to the extent that such work has been cited and acknowledged within the text of my own work. No portion of the work contained in this thesis has been submitted in support of an application for another degree or qualification to this or any other institution.

Signature of candidate: _____

Date: _____

Acknowledgements

This thesis would not have been possible without the support of many people and I would like to thank all those who assisted me in its compilation. I would especially like to thank my husband Michael and my sons Darragh and Conor for their patience and understanding. I also would like to thank my work colleges and fellow students Ben and Joe for the motivation and help they gave me on this journey.

Finally I would like to extend my sincere thanks and appreciation to my supervisor Nigel McKelvey, for his time, effort and guidance in doing this research, without him this would not have been possible.

Abstract

This thesis will demonstrate the effectiveness of processing data for the airline industry using both a MariaDB and MongoDB database. Conclusions will be drawn on pertinent issues such as the performance of processing large datasets. It is difficult to research Big Data without encountering Hadoop as part of the process. Patel et al. (2012. p 1) describe it as the 'optimal solution' in their paper. They conclude that using Hadoop for Big Data processing is a favourable technology to use. However, this dissertation endeavours to research and test a viable alternative that is suitable for enterprise applications. It is for this reason that a MongoDB database, MongoDB and a MariaDB database, MariaDB have been selected for the purposes of this research. This thesis will endeavour to investigate the suitability of the airline industry making use of the data that is available to them which would enable them to specifically target passengers. It is also proposed that the results generated can be used for marketing purposes.

Abbreviations

API	Advanced Passenger Information
BI	Business Intelligence
CSV	Comma Separated Values
DAA	Dublin Airport Authority
DOB	Date of Birth
ISMS	Information Security Management System
ISO	International Organization for Standardisation
MAS	Malaysian Airlines System
OWASP	Open Web Application Security Project
PCI	Payment Card Industry
PNR	Passenger Name Record
SAA	Shannon Airport Authority
QMS	Quality Management System

Table of Contents

1	Introduction	13
1.1	Purpose.....	13
1.2	Background and Objective	13
1.3	Aims and Objective	13
1.3.1	Aims.....	13
1.3.2	Objectives.....	13
1.4	Problem	14
1.5	Hypothesis.....	14
1.6	Outline of Report.....	14
2.	Literature Survey.....	16
2.1	Big Data	16
2.2	Big Data Analytics.....	17
2.3	Big Data and Aviation	20
2.4	MASFlight – Case study	22
2.5	Big Data and Marketing.....	22
2.6	Hadoop.....	23
2.7	MongoDB.....	24
2.8	MariaDB.....	25
2.9	Big Data and Data Protection	26
2.10	ISO.....	26
2.11	Conclusion	27
3	Design.....	28
3.1	Purpose.....	28
3.2	Project Scope.....	28
3.3	Application Design.....	28

3.4	Functional Requirements	29
3.4.1	Class Diagram	29
3.4.2	Sequence Diagram	30
3.4.3	Use Case Diagram	30
3.5	Technical Requirements (Non-functional)	31
3.5.1	Data Scrubbing	31
3.5.2	Databases	32
4	Implementation	35
4.1	The Software Application	35
4.2	Connection to MariaDB database	36
4.3	Connection to MongoDB database	37
4.4	Usability	38
4.5	Data	40
4.5.1	MongoDB Data	40
4.5.2	MariaDB Data	41
5	Testing and Results	43
5.1	The Application Testing	43
5.2	Database Performance Testing	43
5.2.1	MariaDB – Nero Profile Tool	43
5.2.2	MongoDB – mViewer Tool	44
5.3	Results	45
5.3.1	Load Testing	46
5.3.2	Select from Database	50
5.3.3	Application Security	56
5.4	Security	57
5.4.1	SQL Injection prevention	57

5.4.2	Data Security.....	58
5.5	Findings	59
5.5.1	Hypothesis : MongoDB and MariaDB are equally viable options for storing and processing 'Big.	59
5.6	Evaluation.....	61
6	Conclusions and Future Research.....	62
7	References	69
8	Appendices.....	75
	Verify MariaDB connection.....	81
	Connection to MongoDB successful	81

Table of Figures

Table 2.1: Big Data Categories	16
Figure 3.1: Class Diagram.....	29
Figure 3.2: Sequence Diagram	30
Figure 3.3: Use Case	30
Figure 3.1: MariaDB Schema Design	33
Figure 3.2: MongoDB Schema Design.....	33
Table 4.1: Specifications Table.....	36
Figure 4.1: Main GUI	39
Figure 4.2: MariaDB Results.....	39
Table 4.2: MongoDB Data Types.....	41
Table 4.3: MongoDB Collection Details	41
Figure 4.3: MongoDB Collection Details	41
Table 4.4: MariaDB Flight Data Types.....	42
Table 4.5: MariaDB Passenger Data Types	42
Table 4.6: MariaDB Table Details.....	42
Figure 5.1: Nero Profile	44
Figure 5.2: Nero Profile Screenshot.....	44
Figure 5.3: mViewer Graph.....	45
Table 5.1: Queries Per Second Table	46
Figure 5.2: Queries Per Second Graph.....	46
Table 5.3: Testing Conditions.....	47
Table 5.4: Testing Criteria	48
Table 5.5: Test Queries	48
Figure 5.6: Insert – Throughput	49
Figure 5.7: Insert – KB per Sec	50
Figure 5.8: Insert – Avg Responses	50
Table 5.9: Testing Queries	51
Figure 5.10: Select Query 1 - Throughput.....	51
Figure 5.11: Select Query 1 – KB per Sec	52
Table 5.12: Select Query 1 - Avg Responses for all threads	52
Figure 5.13: Select Query 1 – Avg Responses	53

Figure 5.14: Select Query 2 – Throughput.....	54
Table 5.15: Select Query 2 – KB per Sec	54
Figure 5.16: Select Query 2 – KB per Sec	54
Figure 5.17: Select Query 2 – Avg Responses	55
Figure 5.18: Select Query 2 – KB per Sec	55
Figure 5.19: Select Query 2 – MariaDB Select Statements.....	56
Figure 6.1: MongoDB collection example.....	64
Figure 5.1: MariaDB Code analysis	87
Figure 5.2: MongoDB Code analysis	88
Figure 5.3: MariaDB V Mongo Code	88

Code Listings

Code Listing 4.1: Building Sql Query	36
Code Listing 4.2: MongoDB connection.....	37
Code Listing 4.3: Create MongoDB Query	38
Code Listing 4.4: Send MongoDB Query	38

1 Introduction

"Only theory can tell us what to measure and how to interpret it". Albert Einstein

1.1 Purpose

This document is the final dissertation for the M.Sc in Enterprise Application Development. It gives the Software Requirements Specification for this thesis on examining the feasibility for an aviation company to utilize the data in their possession and a comparison of using a MongoDB database and a MariaDB database for this purpose.

1.2 Background and Objective

Big Data is not a new phenomenon; it has in fact been around for some time. However it is only in recent years that technology has been able to process larger than normal datasets. Emerging from this capability to process large volumes of data is the usability of the results. This is something that has become more valuable especially with the growth of social media, which has resulted in vast amount of user information and interactions being stored and used for marketing and targeting potential customers. This thesis endeavours to show how the aviation industry can utilize its data using existing technologies and databases as opposed to implementing a whole new Hadoop structure, using a MariaDB and a MongoDB database for the task.

1.3 Aims and Objective

1.3.1 Aims

- Analyse and evaluate how Big Data can be useful for the Airline industry.
- Analyse and evaluate the difference between MariaDB and MongoDB databases.
- Analyse and evaluate the performance of both databases.

1.3.2 Objectives

- Data sets will be extracted that will demonstrate how statistics can be generated using flight and passenger data.

- Data sets will be extracted that will demonstrate how airlines can use data to generate statistics for marketing purposes.
- The data from both sets will be synthesised to establish if Big Data is a viable technology for the airline industry.
- The performance of both a MariaDB database and MongoDB database will be tested.

1.4 Problem

The aviation industry has always had large volumes of passenger and flight information continually processing through their systems. However due to the quantity of this data and the lack of modern technologies such as Hadoop, being used by aviation companies, these datasets were redundant, as there were no means of easily correlating the information together to generate useful datasets from it. Aviation data hasn't changed much over the years, and the nature of the data is that it won't change a great deal in the future either. It is for this reason that a lot of aviation applications are legacy systems and companies are not always eager to spend money upgrading.

1.5 Hypothesis

MongoDB and MariaDB are equally viable options for storing and processing 'Big Data'

This thesis discusses whether both databases are viable options for an aviation application to implement Big Data analytics. This thesis also examines the performance of each database, when processing the same queries under load.

1.6 Outline of Report

Chapter 2 examines the background of Big Data and its current usage.

Chapter 3 contains design diagrams for the software application.

Chapter 4 details the implementation of the application and the databases used.

Chapter 5 tests the application and compares the databases and analyses the results.
Chapter 6 concludes the thesis and proposes further research.

2. Literature Survey

This chapter explores the current literature and industry practices of Big Data analytics, as well as some of the mechanisms and tools for processing Big Data sets.

2.1 Big Data

Mukherjee et al (2013, p1) describe how Big Data is often broken into four categories, known as the four V's.

Volume	The size of the data being analysed
Velocity	The rate at which the data is being generated
Variety	The different types of data that can exist
Veracity	The uncertainty of the data

Table 2.1: Big Data Categories

Veracity is the most difficult category to manage, but this also makes it the most interesting. Veracity refers to the uncertainty of the data as the data can be out of date, and thus lead to inaccurate results. This can leave businesses with the decision of trusting and using the data or not. Bad data quality costs \$3.1 trillion to the US economy each year. (Tibbetts, H. 2011). Bad data is not something that the airline industry can afford, especially in relation to aircrafts. It is imperative that records relating to the servicing of aircrafts and parts are accurate. Predicting flight arrival and departure times also needs to be accurate for airlines, in order to comply with passenger expectations.

Analysing Big Data correctly is very important. If a company chooses to invest time and resources in gathering and analysing the data that they have, then it is imperative that the outcome of the examination of that data is successful and provides the company with statistics that can be used to enable the company to profitably enhance their business or increase their market share.

Research (Rabi, 2013) has highlighted that the processing of Big Data requires more data processing and storage power than what a company normally has. Scalability is traditionally a required feature when working with Big Data. However, Mukherjee et al (2013), disagree with the common belief that all companies need to invest in larger and more expensive shared storage solutions. It is accepted that global companies such as Google and Facebook require massive scalability in their systems to handle and process the large volume of data that they acquire each day. Mukherjee et al. (2013) don't dispute this, rather they agree completely with this. Their research, involved smaller traditional enterprise businesses that have a moderate amount of Big Data available to them, but not the finances to spend on gathering, sorting and analysing their data. They argue that a 'significant number of deployments of Hadoop in enterprises typically doesn't exceed 16 nodes'. (Mukherjee et al., 2013, p 1). In a study carried out by the company Boeing (Ayhan et al., 2013), they chose to build a data warehouse using IBM's InfoSphere¹ Data Warehouse. This showed that the nature of the aviation data that was being analysed was structured enough that a MongoDB database approach was not required.

2.2 Big Data Analytics

'While it took from the dawn of civilization to 2003 to create 5 Extra bytes of information, we now create that same volume in just two days!' Das and Kumar (2013, p 1). The rate at which Big Data is growing is unpredictable (Murphy, 2014). The emergence of social networking sites and the increase of users posting to the different social media sites, suggests that there is an abundance of "Big Data" floating around in cyberspace. Capturing this data and knowing how best to utilise it is something that most companies still haven't got a comprehensive understanding of.

The Airline IT Trends Survey 2013² reports that only 9% of airlines rated data quality as meeting all their requirements, and only 7% of airlines were reported as having achieved all the necessary integration of different data sources from across their company. (SITA, 2013)

¹ <http://www-01.ibm.com/software/data/infosphere/>

² <http://www.sita.aero/content/airline-it-trends-survey-2013>

It can be suggested that an increasing number of companies are beginning to realise the worth of the large amounts of data that they possess. It is the processing of this data that can be cumbersome for companies. The need for IT specialists is also a challenge for companies interested in utilizing their Big Data. Analysing data can be arduous which is leading to an increase in companies hiring data analyst specialist. Manyika et al (2011), highlighted statistics that imply that there is a need for up to 190,000 more workers with analytical expertise and 1.5 million more data-literate managers and analysts in the United States alone. These statistics infer that in order for a company to delve into Big Data they will have to either hire experts or train existing employees on the new field. A recent report by Forfás³ compiled by the Expert Group on Future Skills Needs⁴ they have described how the Irish government has 'set an ambition for Ireland to become a leading country in Europe in 'Big Data'. This in turn will create major employment opportunities within Ireland. The report then details the actions and recommendations that are required to achieve this.

There are many different perceptions of Big Data Analytics. What one person, may perceive, is not always replicated by another's understanding. Larsen (2013, p 3) defines Big Data Analytics as 'the process of examining large amounts of data, of a variety of types, to uncover hidden patterns, unknown correlations and other useful information'. By this, she is defining Big Data Analytics as a process of analysis, where the objective is to reveal information within the data that is otherwise not obvious or easily accessible in the normal stance. The author believes that this is a common understanding of what Big Data and Big Data Analytics is.

Das and Kumar (2013, p 1), take the definition of Big Data Analytics a step further, they define it as, 'a technology-enabled strategy for gaining richer deeper and more accurate insights into customers, partners and the business and it ultimately gaining competitive advantages'. Their definition defines the reasoning behind Big Data Analytics and what the results of Big Data processing can be used for. Their definition is presumptuous of Larsen's definition and moves forward from that to expand on the reason why companies get

³ <http://www.forfas.ie/>

⁴ <http://www.skillsireland.ie/>

involved in Big Data Analytics. For the aviation industry, there are many aspects to gathering and using big data. The current trend in using aviation data is to enhance the passenger's journey. SITA in conjunction with CAPA India have compiled a white paper entitled 'Transforming the passenger experience'. The paper investigates how IT can improve a passenger's experience within the Indian Air Transport Industry (ATI) by 2015. One of its recommendations is that airlines should exploit the 'goldmine of data that their passenger profiles represent' (CAPA-SITA, 2013). It is their belief that collecting passenger data to increase sales is a major opportunity for the aviation industry.

Larsen's definition is descriptive of what Big Data Analytics is and what it can be used for, while Das and Kumar's definition describes the motive for companies to analyse their data and use the results to target their customers in a way in which their competitors can't. This is because the company will have data relating to each of their customers that they can use.

The above quotes both describe Big Data Analytics in different ways; however for both definitions there is still the issue of how to use this information. In the aviation industry there are many ways in which 'Big Data' can be used. The industry has a large amount of different types of data, which can be used to produce different results. A study by Ayhan et al. (2013) referred to creating a system for analysing aircrafts passing through the US NAS (National Airspace System). Wang et al. (2011) evaluated the efficiency of airports. They devised a method for using data such as passenger throughput and aircraft movement.

Big Data can be separated into two different types, structured or unstructured. The type of data being analysed can affect the speed at which the data can be collected and processed. Structured data is typically data that is organised in some way. Being organised makes it easy to store the data in a normal relational database and also makes searching it an effortless task. An example of such is data which is defined to be of a certain type and can be stored in relational database. Unstructured is the opposite, this is data which is gathered from log files or social media sites and it can contain text and multimedia, it is labelled unstructured as it does not fit neatly into a database. The lack of structure makes gathering and formatting this type of data set a large and resource demanding task. Typically if dealing

with structured data, then implementing a solution using a MariaDB database should suffice. In contrast, unstructured data is more suited to a MongoDB database such as MongoDB.

2.3 Big Data and Aviation

The Airline IT Trends Survey is a joint survey carried out between SITA and airlines. It is currently in its 15th year. The most recent survey is from 2013, it indicates a growth in the need for Business Intelligence (BI) tools for airlines. According to the survey, “Airlines want to know more about their customers and have better information for decision making in their operations.” (SITA, 2013)

Larsen (2013) highlighted the problems facing the aviation industry and the analysis of data. She described the difficulties of analysing aviation data as, ‘The sheer scale of aviation data sources and the proprietary nature of data formats make it difficult to acquire, load and link aviation data’. Aviation data comes from many different sources, and as a result, data sets are typically not uniform, this can make analysis very difficult. Larsen (2013) divides aviation data into the following categories:

- Flight tracking data
- Airport operations data
- Weather conditions
- Airline information
- Market information
- Passenger information,
- Aircraft data
- Air safety reports

For the purpose of this research the author is concentrating on Flight tracking data and passenger information, and how the results of analysing these two groups can be advantageous to the airline industry for the purposes of marketing and targeting passengers. The most simplistic form of data that an airline has is API (Advance Passenger

Information) data. This is essentially the flight manifest. It contains names, date of birth, passport number and other personal information. This data along with details of the flight can be used to orchestrate the billboards and advertisements that passengers arriving at an airport terminal are subjected to. This can include the language that the adverts are written in. Online flight entertainment can also be tailored to suit the passengers on a flight.

Flight information is widely available to the general public, the most common use of this data by the public is to track flights to check if they are going to depart on time or checking the status of a flight that is already in the air, to ensure that it is going to arrive on time.

'Commercially available flight schedule databases provide structured, flat-file data tables containing planned operations for more than 82,000 flights, 350 airlines, and 1,700 airports daily'. (Larsen, 2013)

Passenger flight information is available in different formats. An API (Advanced Passenger Information) message contains details such as passenger name, address, DOB, passport, and Nationality. An API message will contain details of every passenger who is aboard the aircraft as it departs. Another form of passenger information is available in a PNR (Passenger Name Record). A PNR will contain all of the details of the passenger similar to an API, but it will also contain details of the passengers booking, such as when they booked, how they paid, special requirements. A PNR record will typically contain more sensitive data than an API record. For the purpose of this research an API data will be used to correlate passengers and flights. The issue of protection of personal data for this thesis was overcome by creating fictional passenger data. Real passenger data was not allowed to be used for this thesis, for legal and ethical reasons. As the data contained in an API is personal, to use it would have been necessary to get the permission of each passenger, and this was not feasible given the amount of data that was required and the time line for the project.

2.4 MASFlight – Case study

‘masFlight offers a multi-source, integrated aviation operations solution to help airlines, airports, government and other industry stakeholders improve scheduling and operations, recover from delays, reduce cost and identify new market opportunities’ (Crunchbase.com, 2010)

Malaysian Airlines have made great advancements in utilising the flight information available to them. They have created applications that can be used to analyse flight data. MAS (Malaysian Airlines System) Flight ‘is an aviation operations-focused data and analytics platform’ (Masflight.com, 2014). MASFlight offers Software as a Service (SAAS) for the aviation industry to allow different collections of aviation data to be analysed and combined to return informative information. The software was designed by aviation and technology experts who have a knowledgeable understanding of the types and complexity of data that exists in the aviation industry. MASFlight provide solutions for airlines and other aviation companies giving them access to live data such as flight and weather. This information can be used by the companies to assist in planning and management purposes.

The MASFlight database combines data from different sources in a flat table that is searchable and extensible. The advantage of this is that it allows queries to be made against multiple criteria without the use of complex ‘joins’ or relational structures, similar to a MongoDB database. (Masflight.com, 2014)

2.5 Big Data and Marketing

An article published in the magazine Airline Leader (Issue 14, 2012) describes how the airline industry is sitting on a ‘goldmine’ of data. The article outlines how, with current marketing trends the data that airlines have had access to for years, has now become very valuable. This goldmine is commonly called “Big Data”. (Airline Leader, 2012). The article continues to explain how legacy airlines are yet to realise the potential of the data that they can access. The purpose of the report was to highlight how the airlines were overlooking the incalculable potential of retail opportunities that presented itself to the industry. The results

for airlines of analysing their big data sets can reveal patterns and trends that can be used for marketing campaigns to either target a specific set of passengers transiting through an airport, or to target potential customers. The results from analysing the data can be used by the airlines themselves or can be sold on to other interested companies.

‘There are any number of emerging companies whose stock in trade is accumulating and monetising personal data. And data about airline passengers is becoming more and more accessible – and more valuable by the day’, (Airline Leader, 2012). The financial value of passenger data may not be that rewarding for the airlines themselves, but the ability to access this data and pass it on to partner or third-party companies, is a lucrative opportunity for an airline.

Big Data ‘is no longer confined to the realm, of technology. Today it is a Business’ (Schroeck et al, 2012). Companies can proactively, create marketing campaigns to target specific customers. Big Data plays a large part in making this a success for companies. Old style marketing campaigns were focused on promoting a product. In today’s marketing world, the product has changed from being a physical product to an online product. ‘A customer-centric approach can add value to any business and by enabling us to differentiate ourselves from competitors who do not offer the same experience’ (Wood, 2013). It can be suggested that current trends in marketing are leading companies to shift away from the traditional marketing campaigns such as TV and radio adverts. The world of print media has been gradually evolving into digital media over the last number of decades.

2.6 Hadoop

It is evident that Hadoop is a popular database for Big Data applications. Rajjesh and Latha (2013, p 1) describe Hadoop in their title as ‘the Ultimate Solution for Big Data Problems’. The majority of literature on Big Data refers to Hadoop. It is difficult to research Big Data without encountering Hadoop as part of the process. Patel et al. (2012, p 1) describe it as the optimal solution in their paper, which ‘reports the experimental work on big data problem and its optimal solution using Hadoop’. They conclude that using Hadoop for Big

Data processing is a favourable technology to use. Weijia et al. (2012) state that Hadoop is popular for Big Data analysis due to its efficiency at processing data and its ease of set-up for analysts.

2.7 MongoDB

The name MongoDB comes from the word humongous. It is classified as a MongoDB database, however it also has other features. MongoDB can be used as a file system and it has Map Reduce functionality, in its Aggregation component. 'Aggregation groups values from multiple documents together, and can perform a variety of operations on the grouped data to return a single result. MongoDB provides three ways to perform aggregation: the aggregation pipeline, the map-reduce function, and single purpose aggregation methods and commands.' (MongoDB, 2013)

MongoDB is one of the more mature MongoDB databases that are available. Boicea et al. (2012) describe it as 'a class of database management system different from the traditional relational databases in that data is not stored using fixed table schemas. Mainly its purpose is to serve as a database system for huge web-scale applications where they outperform traditional relational databases'. It is for this reason that MongoDB is increasing in popularity, and why it was chosen for this research. It already has many companies such as, MTV Networks, Sourceforge, Forbes, The New York Times and many others using it for their projects (Boicea,et al. 2012).

The emergence of so called 'Big Data' and the need to process it effectively, is one of the reasons that MongoDB and Hadoop have been so popular with companies who want to process their data, this is largely because both components do not require data to be in a structured format. They allow companies to process large volumes of data that have no obvious structure or pattern. This is extremely useful for the type of data that is generated via the internet, tracking user interactions through social media etc. Contrary to this however is passenger and flight data, which is a structured form of data that corresponds to columns and tables in a database. So, therefore the need for a MongoDB database may not

be a primary requirement for analysing this type of data. MongoDB and Hadoop offer analysis and storage which allow companies to create their own solution and use the capabilities of MongoDB and Hadoop to handle the data processing.

2.8 MariaDB

MariaDB is a fork of MariaDB. It has increased in popularity in the last number of year, so much so that companies such as Google, Wikipedia, and RedHat have all switched from Oracle's MariaDB database to using MariaDB (Vaughan-Nichols, 2013). MariaDB is fast becoming the replacement database for companies moving away from MariaDB. The features and partner add-on products that it provides, enables it to be used for harvesting and analysing big data. One of the biggest advantages it has over Oracles' MariaDB is performance (Hutchinson, 2012).

A MariaDB database can be part of the infrastructure for a Big Data application. MariaDB is not a replacement for Hadoop, but it can be used in conjunction with Hadoop. For projects that are processing Terabytes of data, Hadoop is the best choice of software; subsequently the results can then be queried and reported using MariaDB (Zoratti, 2014).

When choosing a database for this thesis, MariaDB was an understandable choice as it is 'a binary-compatible replacement for MariaDB' (Hutchinson, 2012). It is Open Source and based on MariaDB, meaning and the learning curve would be slim. It is also becoming an increasing popular database for companies to use, and is predicated by TechRepublic⁵ to be one of the main open source database for companies 2014 (Wallen, 2014).

⁵ <http://www.techrepublic.com>

2.9 Big Data and Data Protection

The Data Protection Act 1988 (amended 2003) covers the collection, processing and using of personal data. The Data Protection website⁶ defines it as, ‘your fundamental right to privacy. You can access and correct data about yourself. Those who keep data about you have to comply with data protection principles’ (Dataprotection, 2013).

The Data protection act was created to protect the individual and preserve their right to privacy. It was for this reason that the passenger data that was used in this thesis was fictitious. Under the data protection act an individual has the right to access and change any data that is being stored about them. For the purpose of this thesis it would have been unlawful to store any personal data of actual passengers.

An article by Kuner et al. (2012) suggests that ‘Big data poses enormous challenges for data protection’. The article describes how uncovering and analysing large data sets have produced great results in the world of medicine, yet these advances in big data are also testing the data protection laws. ‘Improving user’s experiences is no justification for using consumer information in big data projects, according to Europe’s top data protection officials.’ (Techcentral.ie, 2014).

2.10 ISO

ISO 27001 is the internationally recognised standard for managing the security of information that a company may possess (Certification Europe, 2013). A company that is ISO 27001 accredited signifies that the company is reliable and dependable. It also enhances their reputation, and can reassure their clients that their information is being properly managed by a credible organisation, (Almir, 2013). The current version ISO 27001:2013 which was published on the 1st of October 2013 includes standardised requirements for an information security management system (ISMS). ‘The standard adopts a process approach for establishing, implementing, operating, monitoring, reviewing, maintaining, and

⁶ www.dataprotection.ie

improving your ISMS', (Certification Europe, 2013). Airlines such as Malaysian Airlines, who were awarded the ISO 27001 certification in 2013, have to invest a lot of time and resources into ensuring that they comply with the standards to ensure a high level of security and protection for their data.

Other standards of note are the ISO 9001:2008 standard for Quality Management and the AS9100 the International Standard on QMS specifically developed for aerospace industry. In 2012 the Irish Aviation Authority were re-certified to the ISO 9001:2008 standard for Quality Management for the 15 year in a row (laa.ie, 2012). AS9100 is a widely adopted and standardized QMS for the aerospace industry. This standard is engineering based and is used in the aviation industry to ensure the quality of aircraft and aircraft parts so that companies produce safe and reliable products (Sumranwong, 2011).

2.11 Conclusion

Like most, the airline industry traditionally advertised in print media, TV and Radio adverts. But with the explosion of big data, uncovering a way that airlines can use the vast amounts of data that they have to specifically target current and potential customers is a challenge. If airlines want to engage customers, they need to know as much as possible about the customer. Airlines typically have access to large amounts of passenger personal data, which until recently was never analysed or consolidated with other datasets. The biggest challenge for airlines is how to collect this information. Traditionally aviation data has been stored in separate databases and often in locations, making it an arduous task for companies to correlate passenger information into a useful dataset. MASFlight have made advance in this area, and provide products that can access different live datasets, which aviation companies can use to correlate with their data and analyse the results.

3 Design

This chapter details the system requirements of the software application that accompanies this thesis.

3.1 Purpose

The objective of this thesis is to demonstrate how Big Data can be useful for the Airline industry. This chapter outlines the design of the software application that was developed to show that it is possible to derive information from flight and passenger data.

3.2 Project Scope

The software artefact that was created for this project is a simple Java application that connects to a database to retrieve data. The database contains two tables one which contains flight details and the second contains passenger details.

3.3 Application Design

The GUI was initially to be created using Java Swing, however time constraints on the project guided the developer to focusing on alternatives and discover a plug-in called MigLayout⁷. MigLayout is a layout manager that can be used instead of the Java Swing layout managers; it allows developer to quickly position components on the interface in a linear manner. In this project it will be used to create the interface for the application in a quick and manageable fashion.

Connecting to two different databases means that the application will effectively be divided into two separate applications as the connection to the databases required different code.

⁷ <http://www.miglayout.com/>

3.4 Functional Requirements

Database connection

-The artefact must be able to connect to a database.

Performance

-The performance of the data processing must be measurable.

User Interaction

-The user must be able to select a flight or a destination to analyse.

-The user must be able to specify extra parameters that they want to analyse.

-The results must be visible in a table format.

-The user must be allowed to base a search on different parameters. (such as DOB, Gender, NAT)

-The user should be allowed to organise the results in the table by different columns.

-The user should be allowed to reset the combo boxes

3.4.1 Class Diagram

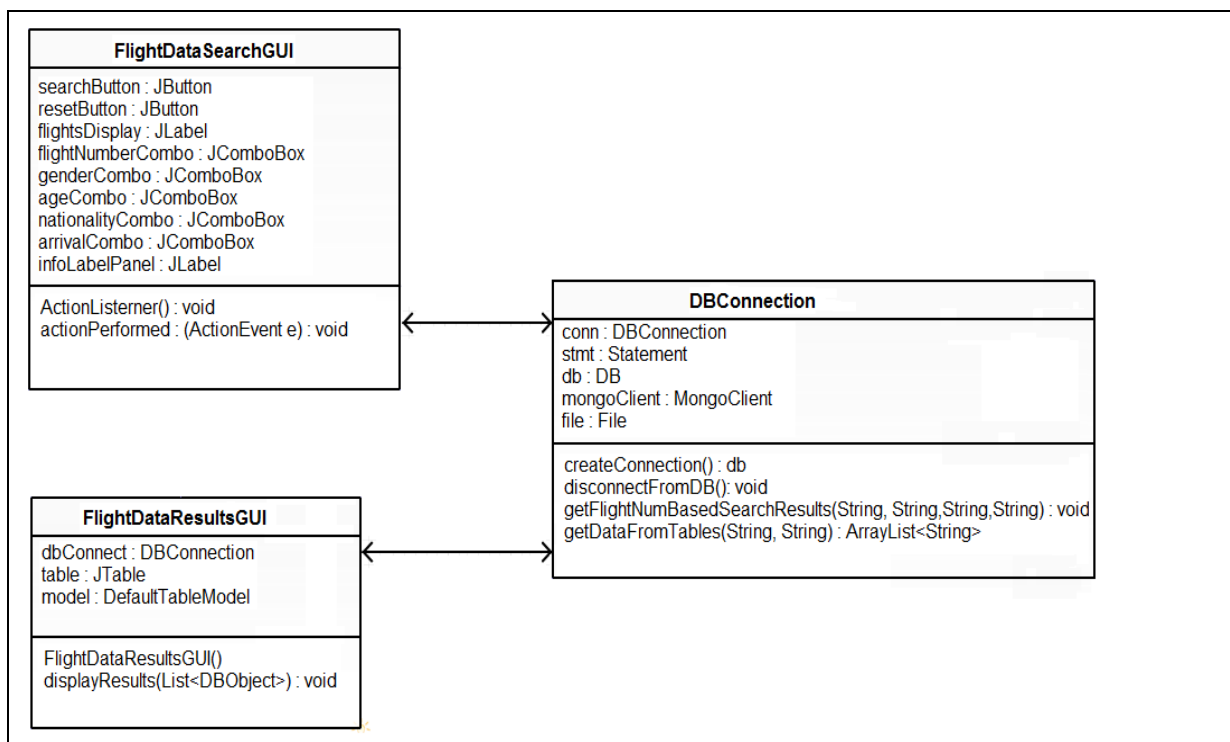


Figure 3.1: Class Diagram

3.4.2 Sequence Diagram

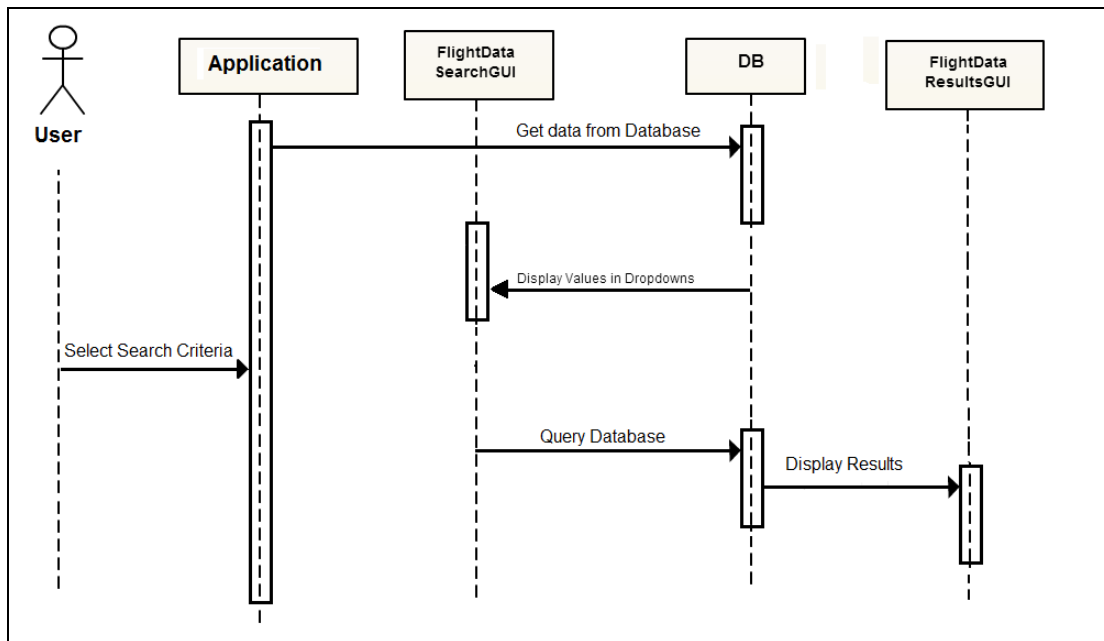


Figure 3.2: Sequence Diagram

3.4.3 Use Case Diagram

On start-up, the application populates the dropdown boxes with the details of the flights from the database. The application allows the user to select the criteria that they want to search by. The search criteria selected by the user, is then used to construct a Select statement that will be used to query the database and the results will be returned and displayed in a table to the user.

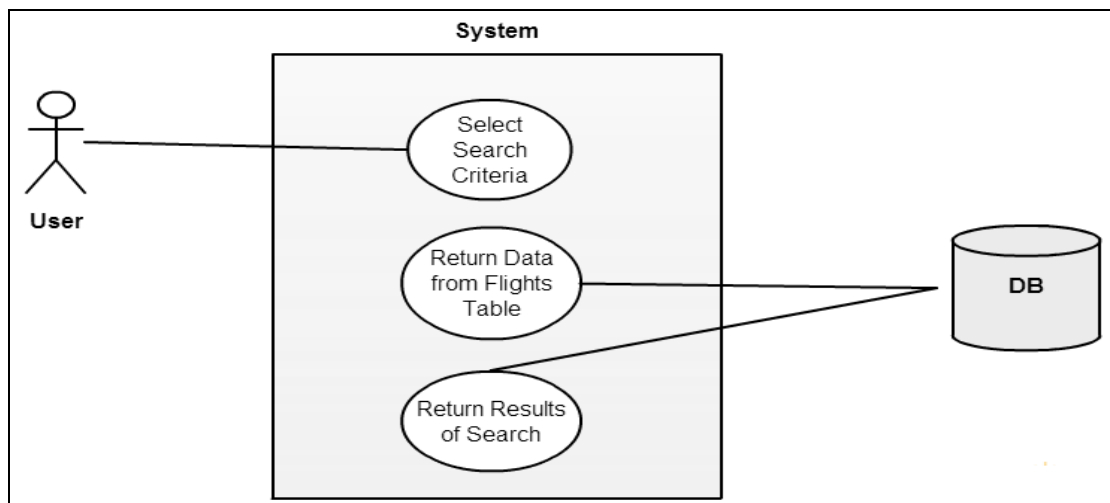


Figure 3.3: Use Case

3.5 Technical Requirements (Non-functional)

3.5.1 Data Scrubbing

Data for this application was gathered from an online resource that accumulates aviation datasets. This website will be used to gather a dataset containing flights from all over the world. The passenger details will be generated manually using fictitious names.

The procedure of cleaning data before use is an integral part of the big data process. It is vital for enhancing the data quality to make it more accurate and provide more consistent results. The data used in this project conformed to a structured format. This was due to the nature of passenger and flight data, which can be easily represented in a spreadsheet. For the purpose of this thesis the dataset didn't warrant an investigation into the tools and techniques of data cleansing. Time constraints forced the author to conduct a manual check on the data and fix any erroneous fields.

The data scrubbing process of the data set used in this study was cleaned manually. The data was initially prearranged using MS Excel and then converted into CSV format and imported into both databases using the tools outlined section 3.3.2. The cleaning process conducted on the passenger data involved scanning the data and removing any user input errors. The data was then verified when it was imported into the database using the 'Import' feature in the HeidiSQL tool. This was because the columns in the tables expected certain values, and if the values in the CSV file that were being imported were not of the correct data type or length, then the import failed. For example the PNR locator was required to be 5 characters in length and had to be unique, also the arrival and departure codes had to be 3 characters long.

A common method of cleaning large data sets is through the use of scripts. Data Scrubbing is a time-consuming activity, but is a necessary step in the process of big data analytics. Scripting languages such as Perl, R and Python can all be used for data scrubbing.

There are issues and challenges with data scrubbing, when cleaning data to remove invalid or duplicate records if there is lack of information on how to ascertain how the record should be corrected, this will result in the record being deleted. This can cause issues, as potentially a large quantity of records could be removed from the dataset. Once data has been cleaned it needs to be maintained, any changes should be reflected in the dataset, it will be necessary to clean the data again, however, only the relevant fields should be cleaned.

3.5.2 Databases

The databases that were chosen for this application were MariaDB, and MongoDB. These databases were chosen as they are both relatively new and are emerging as the databases of choice for a lot of companies processing big data.

3.5.2.1 Schema Design

The importance of a schema design for a database cannot be underestimated. A good design using indexes can have a positive impact on the performance of a database. However, a badly designed schema, where there are too many joins or the join operations are inefficient will have a detrimental effect on the performance. A schema design should strike a balance between read and write performance. Normalisation and Denormalisation can help with the read and write performance.

The schema design for the application connecting to a MariaDB database consisted of two tables. A passenger table and a flights table. The tables were related by the flight number. Every passenger had to have a flight number and each flight number must have an associated flight in the flight table. The layout can be seen in figure 3.4.

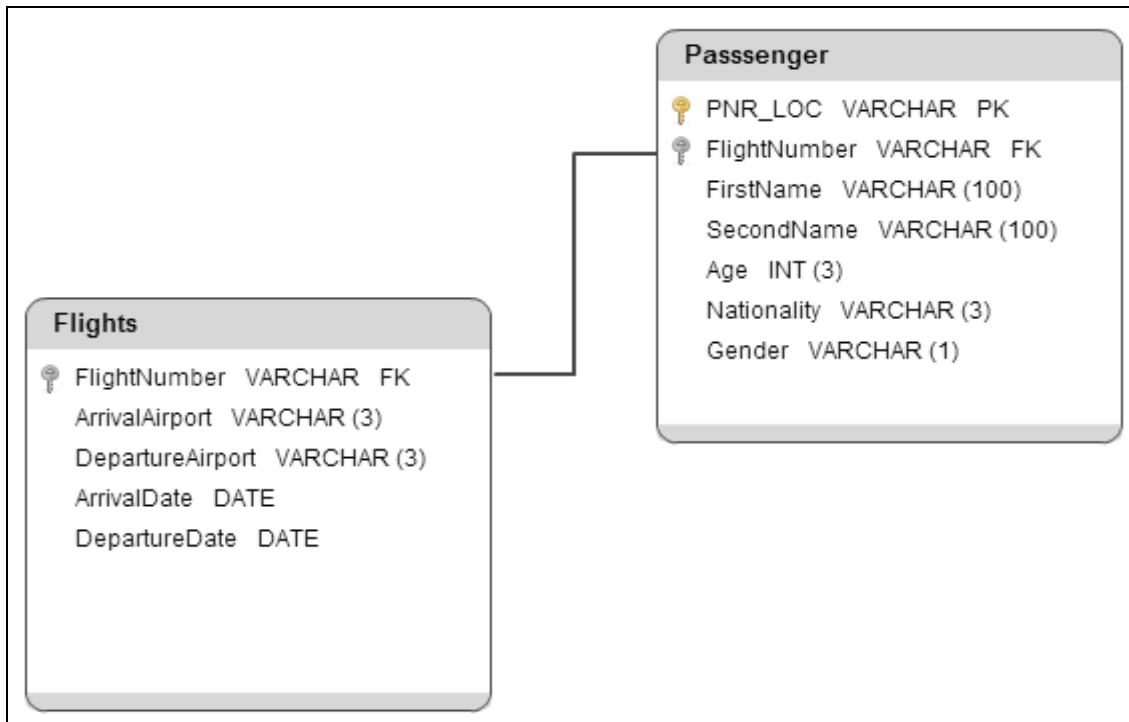


Figure 3.1: MariaDB Schema Design

The application connecting to the MongoDB database consisted of one collection, containing all of the data. The layout can be seen in figure 3.5.



Figure 3.2: MongoDB Schema Design

3.5.2.2 MariaDB

The version of MariaDB that was used for this thesis was MariaDB-5.5. A tool called HeidiSQL was used in the creation of the database tables for this application. HeidiSQL is a windows client that comes packaged with the windows release of MariaDB. The HeidiSQL GUI allows for the import of text and .csv files. For this application, the data was collected and structured using MS Excel and the imported in a .csv format into MariaDB.

3.5.2.3 MongoDB

MongoDB is a MongoDB database. MongoDB refers to its data in terms of collections and documents as opposed to tables and columns in MariaDB. The version that was used for this thesis was 2.4.7. Using MongoDB required download the zip file of the version to be used and extracting the contents. It was relatively quick and easy to implement and there is an abundance of resources available relation to using it. These were factors in choosing MongoDB as the MongoDB database for comparison in this thesis.

For the purpose of this dissertation and allowing for the time constraint, a tool was sourced that could be used with MongoDB. It was for this reason that MongoVUE was downloaded and installed to assist with the research. This is a GUI tool that allows the author to connect to the MongoDB database and view the collections and documents in a GUI format as opposed to running commands from the command line. The Mongo database was started using the 'mongod' command, once the database was running MongoVUE was then used to view the data in the database. This was most useful as it made it easier to visualise the data as it could be seen in a table format, as a JSON object or in a tree view as shown in Appendix B, C and D.

The setting up of the collection in MongoDB was made easy by using the import feature of MongoVUE, which meant that the data, which was already in MariaDB format, could be easily imported into MongoDB. Having to manually create a collection from scratch and add in all of the documents could be an arduous task if doing it all through the command line console.

4 Implementation

This chapter details how some of the various elements described in the design chapter were implemented.

4.1 The Software Application

A software application was created to connect to a database and retrieve data. The application allowed the user to base their search on either a flight number or an airport code. Other attributes could then be selected to refine the search. Based on the search criteria, the application could then construct a query either in SQL or JSON format, (depending on the database being used). The initial testing phase included exploratory testing to ensure that the application functioned appropriately and to eradicate any bugs. This was an ongoing process throughout the entire development of the application.

Figure XX below details the spec of the machine upon which the application and databases were installed.

Hardware	
Manufacture	Samsung Electronics
Model	S3511
Processor	Intel(R) Pentium(R)
OS	Windows 7
RAM	4GB
System Type	64-bit OS
MariaDB	
Version	5.5
Max Blob/Clob Size	4GB
Max Table Size	256TB
Max Column Name Size	255
Max Connections	10000
Max Threads	12000
MongoDB	

Version	2.4.7
Max Connections	20000
Max Threads	12000
file size	Unlimited
cpu time	Unlimited
virtual memory	Unlimited
open files	64000
memory size	unlimited
processes/threads	64000

Table 4.1: Specifications Table

4.2 Connection to MariaDB database

The testing of the MariaDB code that retrieved data from the database was based on different combinations of data being selected from the drop down boxes and used to construct the select statement. It was for this reason that it was decided to narrow the scope of the application and force the user to select either a flight number or an arrival airport to base their search on. This made it simpler to construct an SQL Select statement in the code by using a string builder to construct the query. A regular expression was used to determine if the base value passed in was in the format of a flight number or not. Refer to code listing 4.1 below.

```
//use a string builder to build up the query
StringBuilder sb = new StringBuilder();

if (flightNumber_ArrivalAirport != "ALL") {
    if (flightNumber_ArrivalAirport.matches(".*\\d+.*")) {
        sb.append("SELECT * FROM passengers WHERE FlightNumber = '" + flightNumber_ArrivalAirport + "'");
    } else
        sb.append("SELECT passengers.*, flights.ArrivalAirport FROM passengers, flights WHERE ArrivalAirport = '"
            + flightNumber_ArrivalAirport + "'");
} else sb.append("SELECT passengers.*, flights.ArrivalAirport FROM passengers, flights");
```

Code Listing 4.1: Building Sql Query

Other search criteria were then appended to the select query if they didn't have the 'ALL' value passed in from the user. SQL injection attacks are always a concern when developing applications that have a database connection. The application was designed using

dropdown boxes populated from the database; this limited the user to only selecting data that was in the lists provided.

4.3 Connection to MongoDB database

There was a learning curve with the MongoDB database, as a result more time was spent on getting the application to connect to the database and successfully return data from the collections. To connect to MongoDB required downloading a mongo java driver and importing it into the DBConnection class. This allowed for an instance of MongoClient to be created and this was then used to connect to the database. Refer to code listing 4.2 below.

```
DB db = null;
MongoClient mongoClient = null;

/*
 * Connect to MongoDB
 * @throws IOException, ClassNotFoundException, SQLException
 * @returns MongoDB connection
 */
public DB createConnection() throws IOException, ClassNotFoundException, SQLException {
    //create a new mongoDB connection
    mongoClient = new MongoClient("localhost", 27017);
    db = mongoClient.getDB("JoinedCollection"); //get the collection from the database
    Set<String> colls = db.getCollectionNames();

    for (String s : colls) {
        System.out.println(s);
    }
    return db;
}
```

Code Listing 4.2: MongoDB connection

Once the connection had been established the application was able to connect and return the data from the JoinedCollection.

Creating the search query for Mongo was more complicated than creating the search string for MariaDB. The Mongo query involved creating a BasicDBObject to store the query in and also creating a list of DBObjects to store the criteria for the search. The objects were added to the search criteria using the 'and' operator. (Code Listing 4.3)

```
searchQuery.put("$and", obj);
```

Code Listing 4.3: Create MongoDB Query

The query was then sent to the database and the results were returned as a list of DBObjects. (Code Listing 4.4)

```
//Add the results of the search to a list of Database objects  
List<DBObject> resultsList = collection.find(searchQuery).toArray();
```

Code Listing 4.4: Send MongoDB Query

4.4 Usability

The software application was based on a very simple idea. It will allowed a user, to query the database and return results from that database. The reason for the application was for testing purposes to provide an analysis of two databases that could be used as alternatives to Hadoop for processing big-data.

When the application was launched the user was presented with the GUI as shown in Fig 4.1. The user then selected the criteria that they require to search on. They must select arrival airport or flight number, and then optionally select values from the Gender, Age and Nationality dropdown boxes.

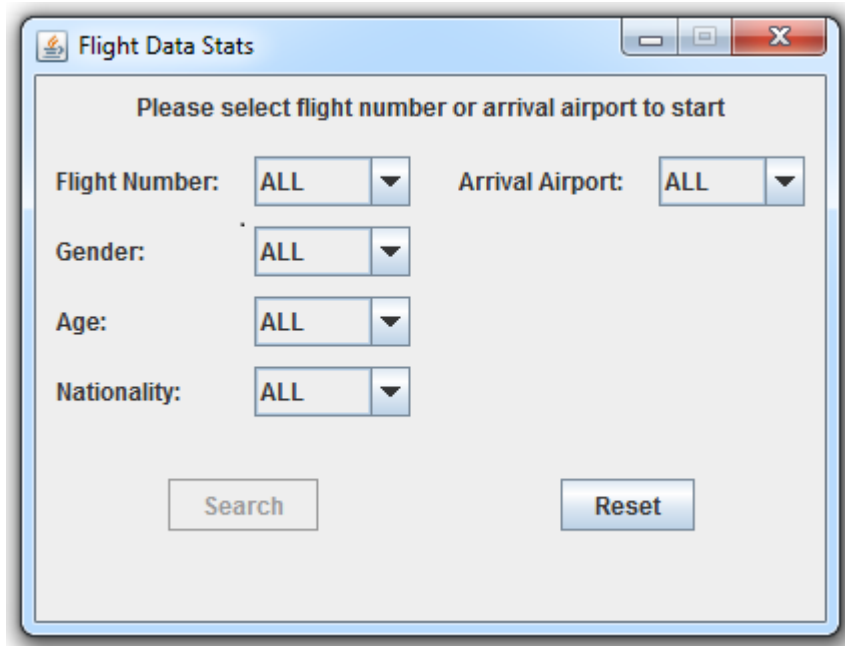


Figure 4.1: Main GUI

Once they had selected their search criteria, they had click the search button to send their query to the database and return search results. Fig 4.2 below shows the results that are returned from MariaDB when selecting all passengers who have a destination (arrival airport) of ABJ

PNR_LOC	FlightNumber	FirstName	SecondName	Age	Nationality	Gender	ArrivalAirport
AAEOI	3J552	Bula	Magar	49	FIN	M	ABJ
AGJWO	2B410	Jean	Severns	3	NLD	M	ABJ
AHFIE	SP442	Marya	Burnett	63	MLT	M	ABJ
AOWOW	V7151	Masako	Lotz	94	JPN	M	ABJ
AQOPE	J8150	Elna	Seery	82	TZA	M	ABJ
AQWXC	J8150	Walker	Martin	13	AFG	M	ABJ
ASDRY	4D442	Tegan	Chattin	19	MMR	M	ABJ
BAAQK	G3179	Christin	Haydel	60	THA	M	ABJ
BBHHH	J8150	Griffin	Janice	41	COG	M	ABJ
BCNDJ	4D442	Brannan	Genevive	45	YEM	M	ABJ
BGHHG	UO277	Emmy	Bucklew	46	CZE	M	ABJ
BGHRW	J8150	Collins	Phillip	16	CAN	M	ABJ
BGYHF	J8150	Lawrence	Boeck	4	NZL	M	ABJ
BHEWP	4D442	Probert	Paula	5	AND	M	ABJ
BIOHO	4D442	Nixon	Jessica	51	USA	M	ABJ
BISOG	4D442	Kleiner	Erlene	98	TJK	M	ABJ
BJEIB	J8150	Cornish	Cristin	88	TZA	M	ABJ
BJUIV	J8150	Selene	Fee	100	FJI	M	ABJ
BKLFD	4D442	Wyatt	Briganti	15	USA	M	ABJ
BMBMW	G3179	Anthony	Wilson	45	MKD	M	ABJ
BMNVM	HA268	Terry	Davis	98	YEM	M	ABJ
BNFDK	J8150	Josue	Horst	38	NLD	M	ABJ
BNULU	J8150	Thompson	Helen	24	COG	M	ABJ

Figure 4.2: MariaDB Results

4.5 Data

The flight data for this application was obtained through a dataset provider called openFlights.org⁸. The passenger data was generated by the author and contained fictional information. The data set that was used for both the MariaDB and MongoDB database was identical, only the structure and storage of the data differed. The data was correlated into a comma separate value (CSV) file. For the MariaDB project this data was in two separate files and it was then imported via HeidiSQL into two tables for MariaDB. The MongoDB data was combined into one CSV file and was then imported into one collection in MongoDB using MongoVUE⁹.

4.5.1 MongoDB Data

The MongoDB data was stored in a single collection in MongoDB. Creation of the MongoDB collection allowed for more flexibility compared to MariaDB. The classification of each field with an appropriate date type and field size was not a consideration for MongoDB, as it allowed for flexibility and for different values to be stored in a corresponding field in the same documents. The data consisted of the following fields:

Field	Data Type
PNR Locator	STRING
Flight Number	STRING
First Name	STRING
Second Name	STRING
Age	INT32
Nationality	STRING
Gender	STRING
Departure Airport	STRING
Arrival Airport	STRING
Departure Date	STRING

⁸ <http://openflights.org/data.html>

⁹ <http://openflights.org/data.html#route>

Arrival Date	STRING
--------------	--------

Table 4.1: MongoDB Data Types

The MongoDB data originally started out as two collections and through the course of testing they were combined into one collection to fit in with the model of a MongoDB database. Figure 4.2 below details the number of documents in the collection and the size of the collection.

Collection	No. Documents	Size (kb)	Avg. Doc Size (kb)
flights_passengers	3672	1004	.0273

Table 4.2: MongoDB Collection Details

These results were obtained by running the dbStats command demonstrated in figure 4.3, on the collection in the database.

```
db.flights_passengers.stats(1024)
```

Figure 4.3: MongoDB Collection Details

4.5.2 MariaDB Data

The MariaDB data was stored in two relational tables called flightData and passengers. As an RDBMS, MariaDB permits the fields to be declared of a particular data type and size. Using fixed size fields, ensured that the data being entered into these fields could be validated to ensure that they are of the correct format and do not exceed the allowed size. Using the appropriate data type and size also had a positive impact on the performance of the database as it will be using as much memory as is necessary when inserting and retrieving data. The flight data consisted of the following fields and data types.

Field	Data Type
Flight Number	VARCHAR(7)
Arrival Airport	VARCHAR(3)

Departure Airport	VARCHAR(3)
Arrival Date	DATE
Departure Date	DATE

Table 4.4: MariaDB Flight Data Types

The passenger data consisted of the following fields data types:

Field	Data Type
PNR Locator	VARCHAR(5)
Flight Number	VARCHAR(7)
First Name	VARCHAR(50)
Second Name	VARCHAR(50)
Age	INT(2)
Nationality	VARCHAR(3)
Gender	CHAR(10)

Table 4.5: MariaDB Passenger Data Types

Figure 4.5 below details the number of rows and the size of the data in each table.

Table	No. Rows	Size (kb)
Flights	385	48
Passengers	2029	304

Table 4.6: MariaDB Table Details

5 Testing and Results

This chapter outlines the testing that was conducted for this thesis and also contains an evaluation of the results.

5.1 The Application Testing

Testing of the completed application was necessary to gather results pertaining to the performance of the application and to ascertain that it could return results from both databases. The majority of this testing was manual testing. The test steps/plans for this can be seen in appendix E.

5.2 Database Performance Testing

This section details the performance testing that was conducted on the databases that were used in this thesis. The initial testing phase involved sourcing software that could connect to each database while the application was running and test its capabilities. The software tools that was chosen for the MariaDB database was 'Nero Profile Tool' and mViewer was chosen for the MongoDB database. Two different tools were chosen as it was difficult to get a tool that could easily be run on a Windows OS which could monitor the processing of each database. Using two different tools also demonstrated the support that is available for these databases. The second phase of testing was to load test each database, this was carried out was done using JMeter, which allowed a test to run the same query against each database under different loads and analyse the results.

5.2.1 MariaDB – Nero Profile Tool

Performance testing was conducted on the MariaDB database using a tool called 'Nero Profile'¹⁰. Nero Profile is a freeware tool that was positioned between the application and the database, as depicted in figure 5.1 below. It captured all requests from the application

¹⁰ *<http://www.profilesql.com/>

to the database. It can record the number of queries per second and the number of rows returned.



Figure 5.1: Nero Profile

“The Nero Profile SQL is an application that intercepts all the queries from the client and works as a proxy server. This technology allows you to take control of the access to the database and to identify the bottlenecks in queries.” (Profilesql.com, 2014). Figure 5.2 below details the queries per second.

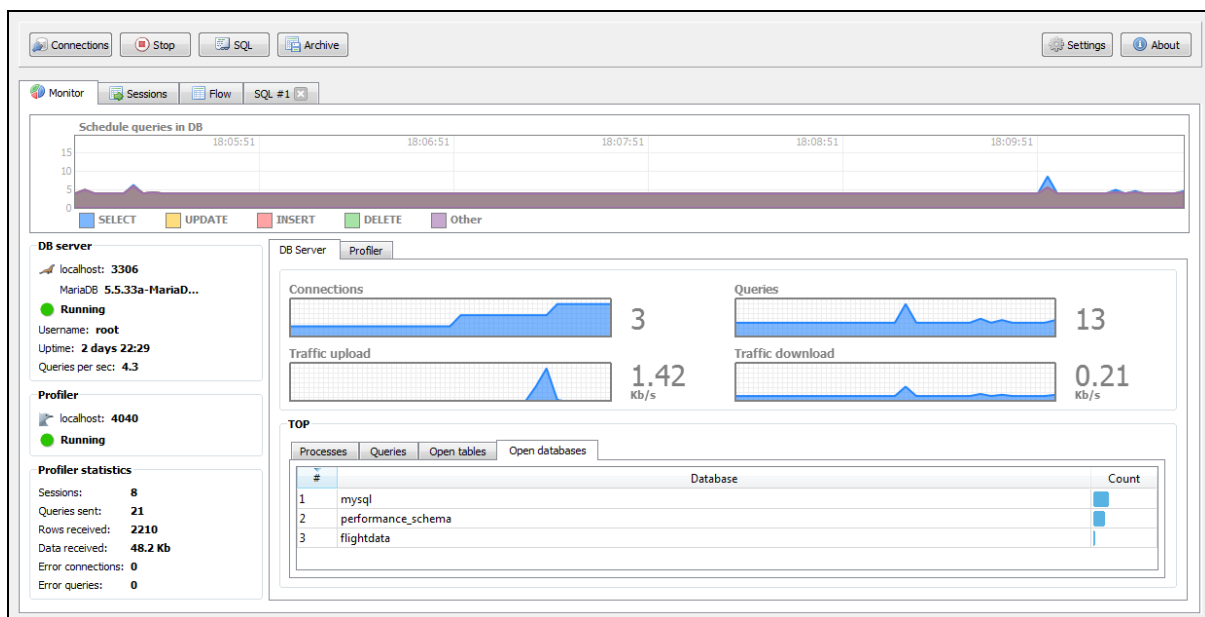


Figure 5.2: Nero Profile Screenshot

5.2.2 MongoDB – mViewer Tool

A tool called ‘mViewer’¹¹ was used to conduct performance testing on the MongoDB database. mViewer is a freeware tool that can be used to manage MongoDB via a GUI. mViewer contains a war file which is deployed on start-up. The package contains a start-up

¹¹ <http://imaginea.github.io/mViewer/0.9.2/>

script, so when mViewer has started, opening a link to the 'localhost'¹² in a web-browser provided access the GUI of the tool.

Figure 5.3 below was generated using mViewer, details the number of queries per second that MongoDB processed when the application was initially loading in the data to the drop down boxes and the number of queries per second that were processed when a full search was done for all entries in the collection graph.

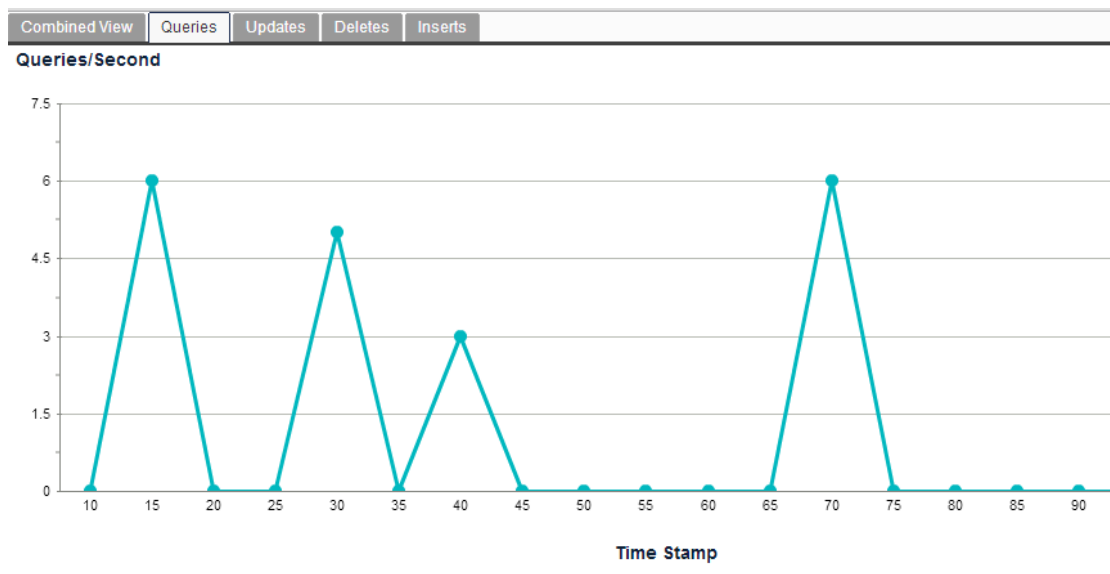


Figure 5.3: mViewer Graph

5.3 Results

The following section details results of the comparisons of the data retrieval between MariaDB and MongDB. The categories for comparison are, the number of Queries Per Sec, the number of sessions, the number of Queries sent, the number of rows received and the amount of Data Received (kb).

¹² <http://localhost:8080/index.html>

	Queries Per Sec
MariaDB	4.3
MongDB	6

Table 5.1: Queries Per Second Table

On a search MongoDB was able to return more queries per second than MariaDB. This is illustrated in figure 5.1 below.

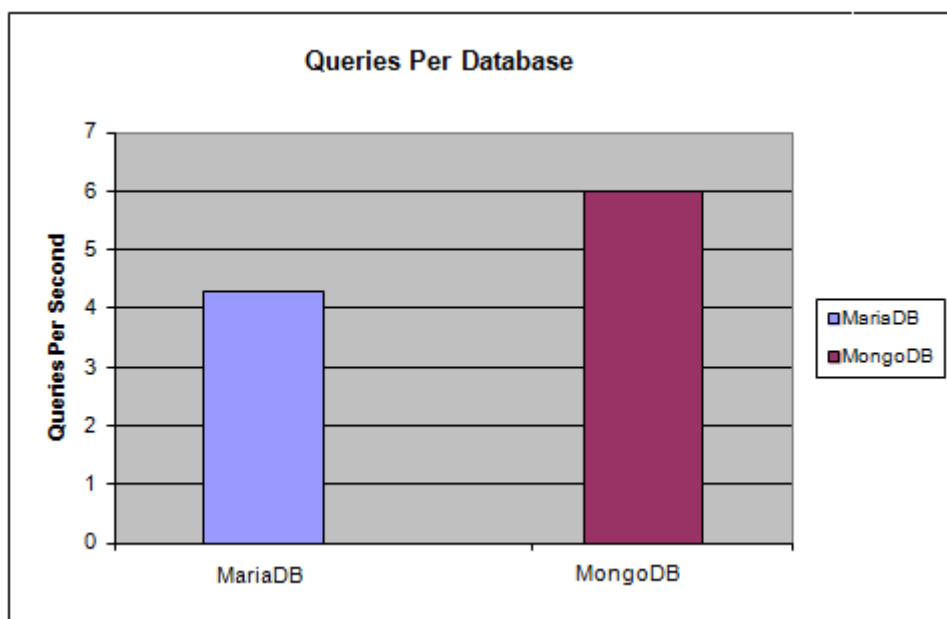


Figure 5.2: Queries Per Second Graph

5.3.1 Load Testing

Enhanced testing was implemented by running a load test on each database. This testing was facilitated with the use of a tool called JMeter. JMeter is a load-testing tool that is primarily used for testing web services, however it can also be used to test objects such as a database. It was important to be consistent when performing a load test and its ability to run MongoDB commands in addition to SQL commands was an important reason for choosing JMeter. It also facilitated a quick connection to each database and allowed the user to construct test plans with could run different queries. JMeter also provided the

facility to specify the number of threads to be issued for a query and the up time between each thread. Using the same tool for testing of each database was important as it allowed for equality and derived accurate comparisons. Load test were carried out using different increments of users from 100 to 5000. The testing was all completed using JMeter. The following are the conditions that were used for each test:

Total Number of Threads	100	200	600	1000	3000	5000
Number of users	100	200	600	1000	1000	1000
Interval between users	10	10	10	10	10	10
Repeated	1	1	1	1	3	5

Table 5.3: Testing Conditions

For the purpose of this thesis, tests were carried out on each database using different number of threads (users) connecting to it and issuing the same command. The graphs in the following sections depict the differences that were recorded between the two Databases with the load tests. The testing was split into two sections, testing by means of inserting data into a table and testing by means of selecting data from a table. The following are the criteria that were measured with each test.

Test	Description
Throughput (Requests per second)	This is a measure of the number of transactions per second that can be copied to and from the databases. It will measure the number of users' interactions that can be handled.
Kilobytes per second	This is a measurement of the throughput in Kilobytes per second. It is a measure of the bandwidth consumption in kilobuyes per second that is generated by the test.

Average number of responses	This result is generated by dividing the number of users by the response time. This is a measurement of the number of responses from the database per user.
-----------------------------	---

Table 5.4: Testing Criteria

5.3.1.1 Insert into Database

To test each database on the ability to insert data, test plans were created with the following queries.

	MariaDB	MongoDB
Query 1	<pre>INSERT into passengers_copy (PNR_LOC, FlightNumber, FirstName, SecondName, Age, Nationality, Gender) VALUES ('QWERT', 'AC1234', 'Joe', 'Bloggs', 21, 'IRE', 'M'); INSERT into flights_copy (FlightNumber, DepartureAirport, ArrivalAirport, DepDateTime, ArrDateTime) VALUES ('AC1234', 'DUB', 'SYD', '2014-01-11', '2014-01- 13');</pre>	<pre>db.flights_passengers_copy.insert({PNR_LOC: "QWERT", FlightNumber: "AC2356", FirstName: "Joe", SecondName: "Bloggs", Age: 21, Nationality: "IRE", Gender: "M", ArrivalAirport: "NBO", DepartureAirport: "MGQ", DepartureDate: "11/01/2014", ArrivalDate: "12/01/2014" });</pre>

Table 5.5: Test Queries

5.3.1.2 Throughput

There was little difference to be seen when inserting a record into each database, however as the number of threads increased, the number of requests per second also increased. Consequently it could be seen that the number of requests per second for the MariaDB database began to exceed the MongoDB database. This shows that when inserting data the MariaDB database was handling more requests per second than the MongoDB database.

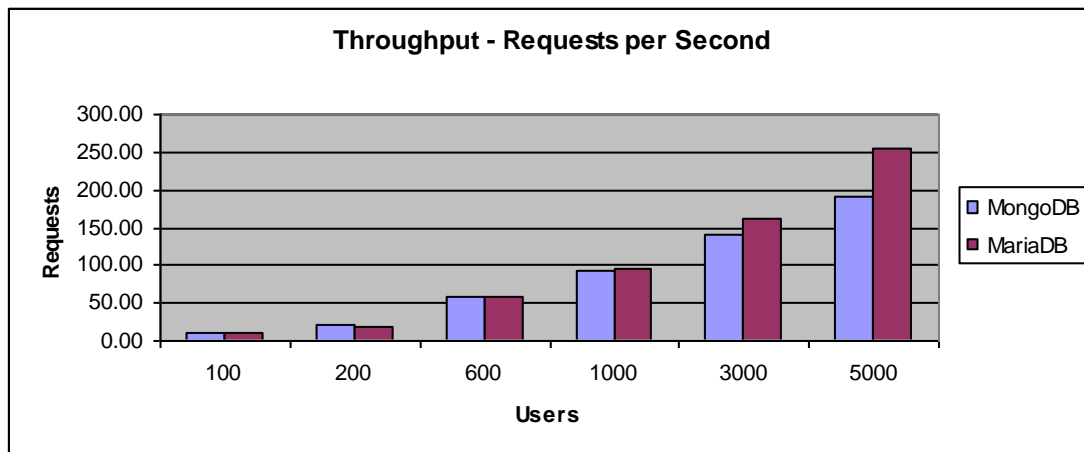


Figure 5.6: Insert – Throughput

5.3.1.3 Kilobytes per Second

The kilobytes per second were also measured for each query. The kilobytes per second test tested the ability of each database to process large volumes of data quickly. This test was important as it gave an indication as to the performance of each database. When the insert query was being run; the MariaDB database was consistently processing more kb per second than the MongoDB database when the load was minimal. However as the load increased as expected each data base processed more Kb per second, however, again the MariaDB database consistently processed a greater amount than the MongoDB database.

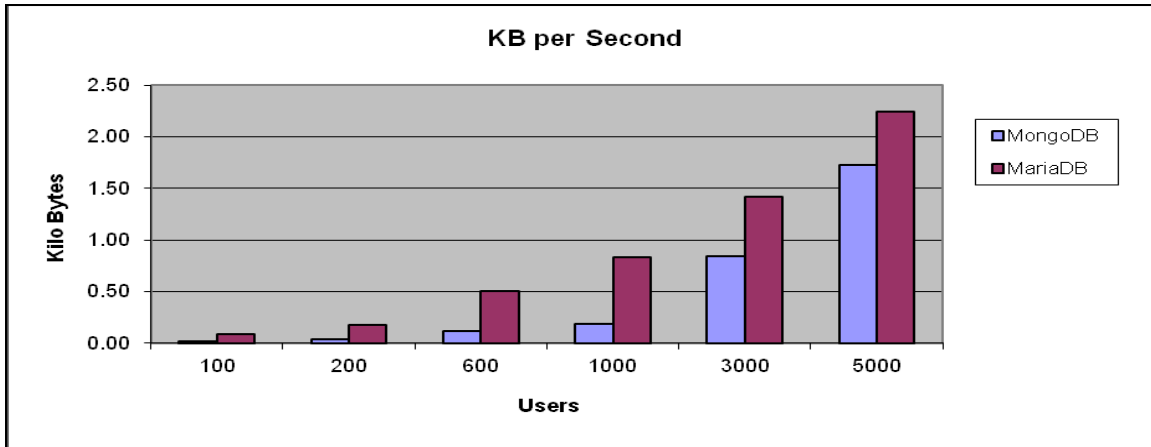


Figure 5.7: Insert – KB per Sec

5.3.1.4 Average Responses

The average number of responses which is the arithmetic mean for all responses (sum of all times / count) was also recorded for each test. As can be seen from figure 5.4 below the average number of responses for the MongoDB database was greater than the MariaDB database.

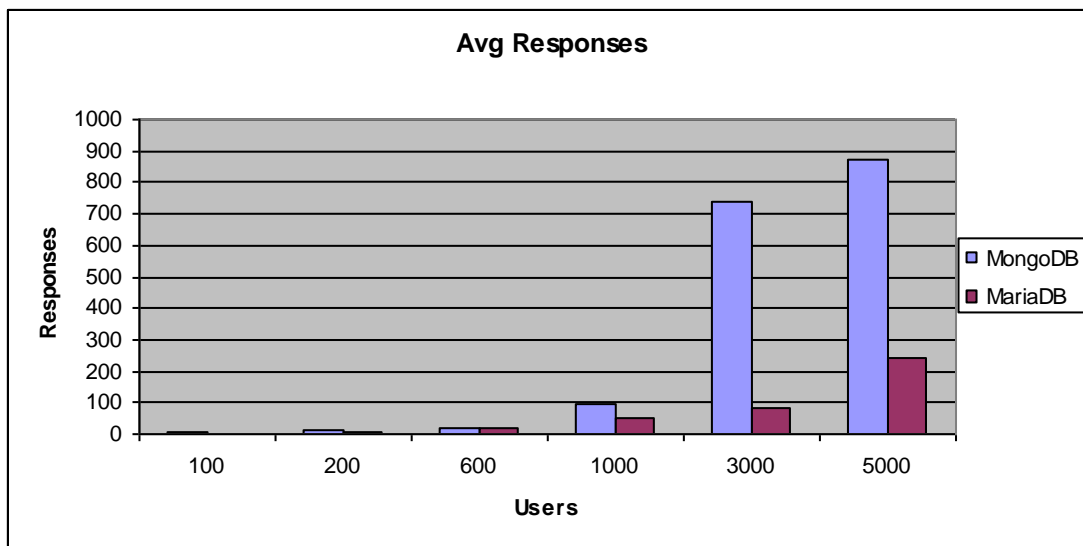


Figure 5.8: Insert – Avg Responses

5.3.2 Select from Database

In addition to inserting data, tests were carried out that involved selecting data from each database and the results were compared. Test plans were created for each database, and the following queries were run:

	MariaDB	MongoDB
Query 1	SELECT * FROM passengers where Gender ='M';	db.collection.find("Gender", "M");
Query 2	SELECT * FROM passengers INNER JOIN flights ON passengers.FlightNumber=flights.FlightNumber;	db.collection.find();

Table 5.9: Testing Queries

5.3.2.1 Query 1 - Throughput

The purpose of the first select query that was run was to return all record where the Gender was Male. The graph below shows the difference in the throughput measured for each database.

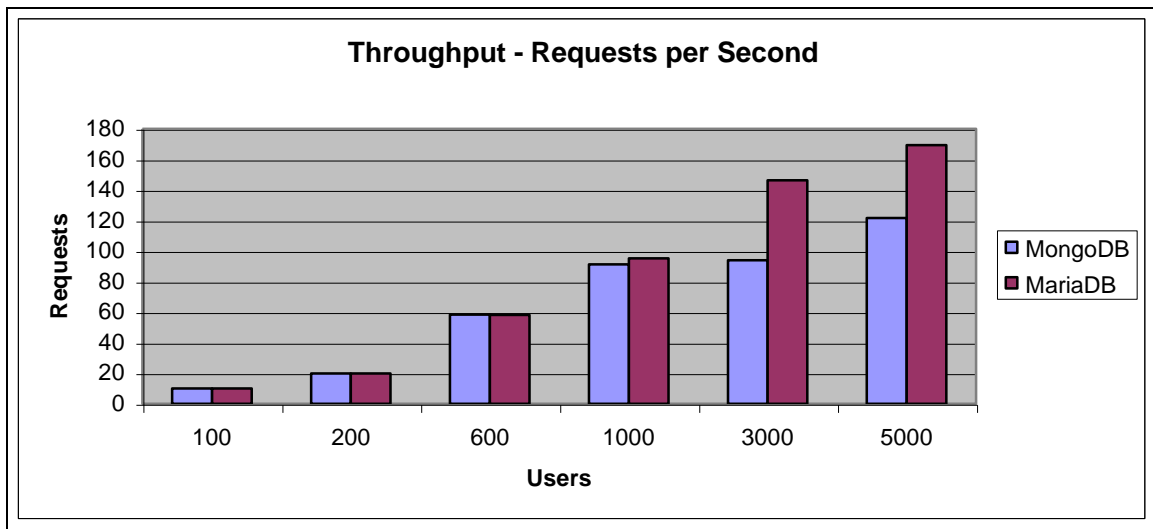


Figure 5.10: Select Query 1 - Throughput

From the above graph it can clearly be seen that when using a low number of users, both databases were performing on a par, however as the number of users increased it can be seen that the MariaDB database was able to process more requests per second.

5.3.2.2 Query 1 - Kilobytes per Second

Figure 5.6 below illustrates the number KB per second for each database when executing the first query. The MongoDB database, MongoDB consistently processed at a higher rate than the MariaDB, MariaDB.

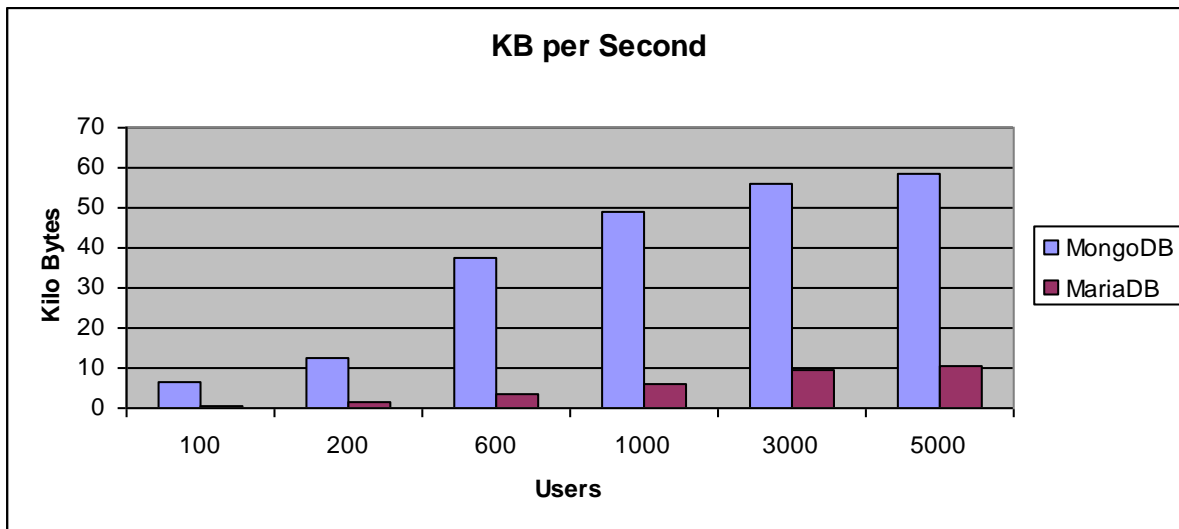


Figure 5.11: Select Query 1 – KB per Sec

5.3.2.3 Query 1 - Average Responses

When measuring the number of average responses for the first query, MongoDB was also far superior to MariaDB. Figure 5.7 below shows the number of responses for load test. MongoDB was outperforming MariaDB for each test.

	100	200	600	1000	3000	5000
MongoDB	8	9	20	55	1246	1512
MariaDB	6	4	6	30	324	369

Table 5.12: Select Query 1 - Avg Responses for all threads

This performance difference dramatically increased as the number of threads increased, as is illustrated in figure 5.8 below. The initial number of responses was small thus there was little difference between the two databases, but as the number of threads increased into the thousands, so too did the average number of responses for MongoDB. The average number of responses for MariaDB also increased, but not as much as MongoDB.

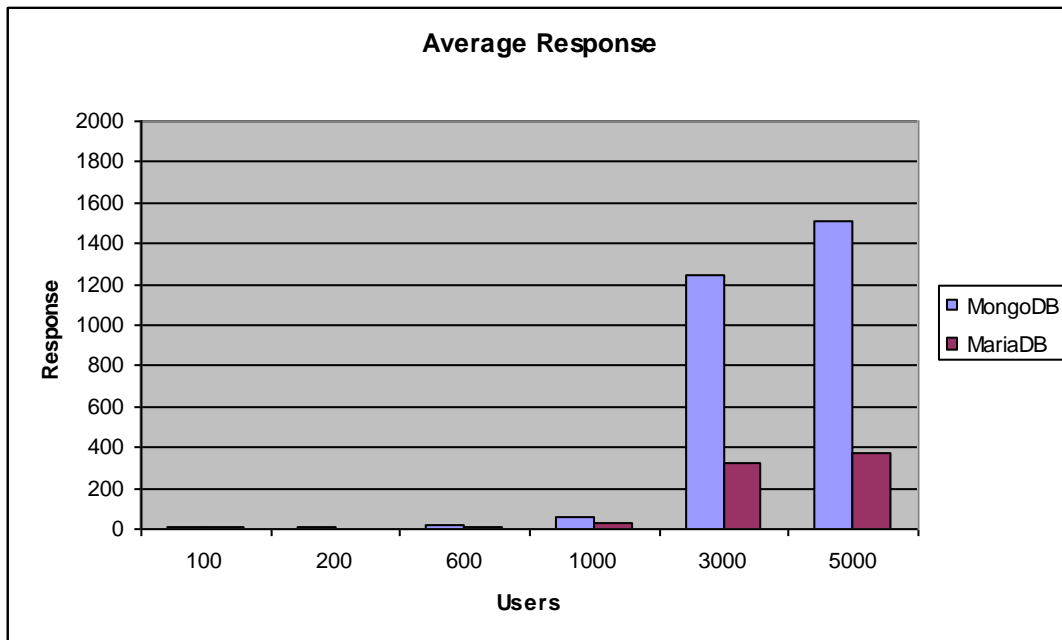


Figure 5.13: Select Query 1 – Avg Responses

5.3.2.4 Query 2 – Throughput

The second select query that was executed on each database was a request to return all data from the databases. The graph below shows the difference in the throughput measured for each database when this query was performed.

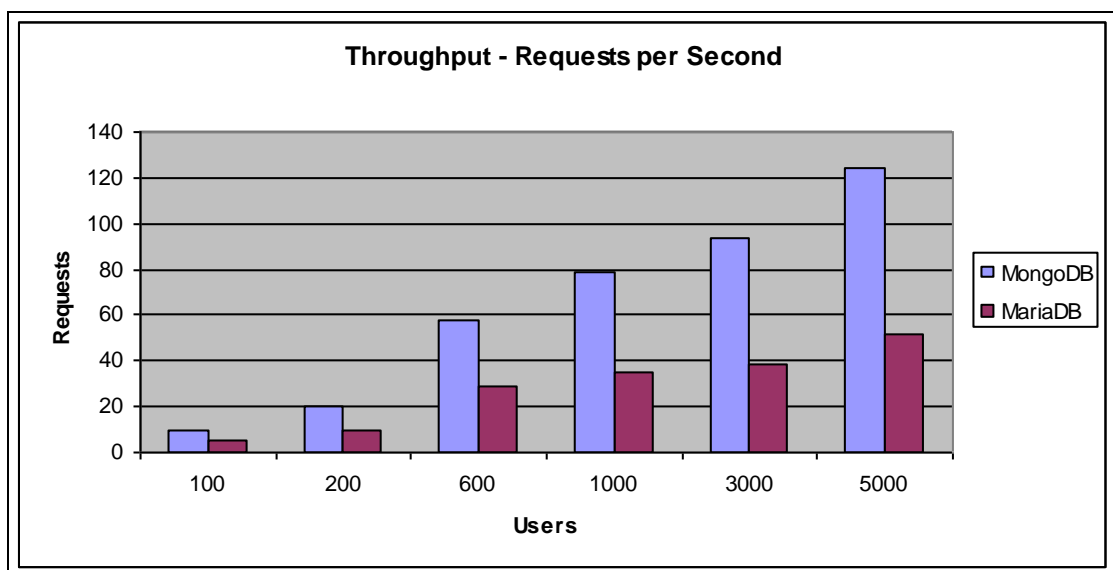


Figure 5.14: Select Query 2 – Throughput

From the above graph it can clearly be seen that when selecting all of the data from the databases MongoDB performs stronger than MariaDB. This was an expected result, for the reason that the MariaDB database is required retrieve data from more than one table combine results, the alternative for the MongoDB solution is that all of the data is contained in one document.

5.3.2.5 Query 2 - Kilobytes per Second

Figure 5.10 below illustrates the number KB per second that each database processed. For this query, MongoDB was more efficient in its performance of the query.

	100	200	600	1000	3000	5000
MongoDB	6.31	12.45	36.3	46.82	49.64	51.41
MariaDB	666.7	1322.1	3766.8	4663.9	5109	6818.2

Table 5.15: Select Query 2 – KB per Sec

Figure 5.11 illustrates the data from the table in figure 5.10, it can be seen that the levels of MongoDB do not register on the graph in comparison with MariaDB.

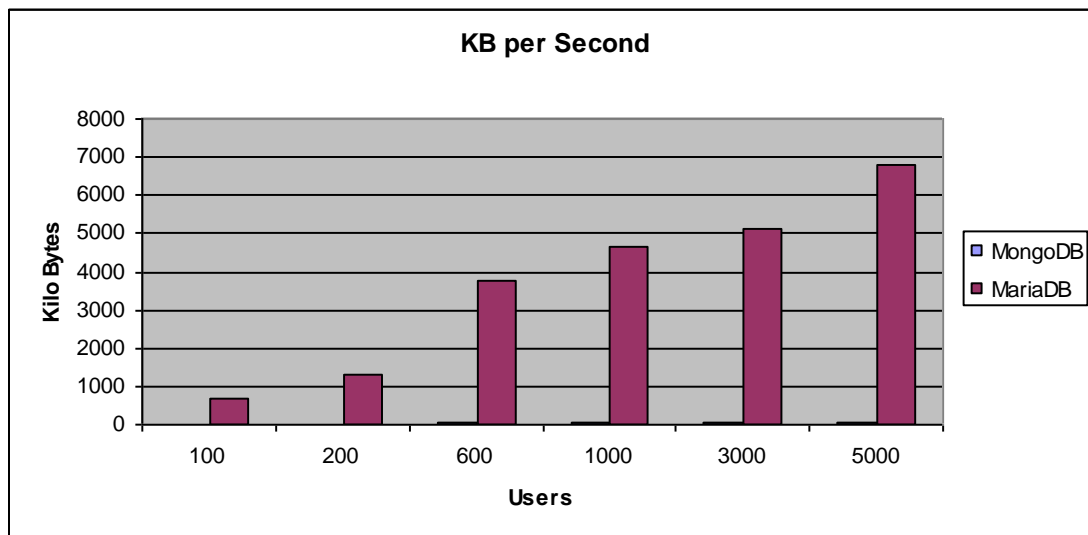


Figure 5.16: Select Query 2 – KB per Sec

5.3.2.6 Query 2 - Average Responses

When measuring the number of average responses for the select all query, both databases were equally consistent however this changed as the number of users increased.

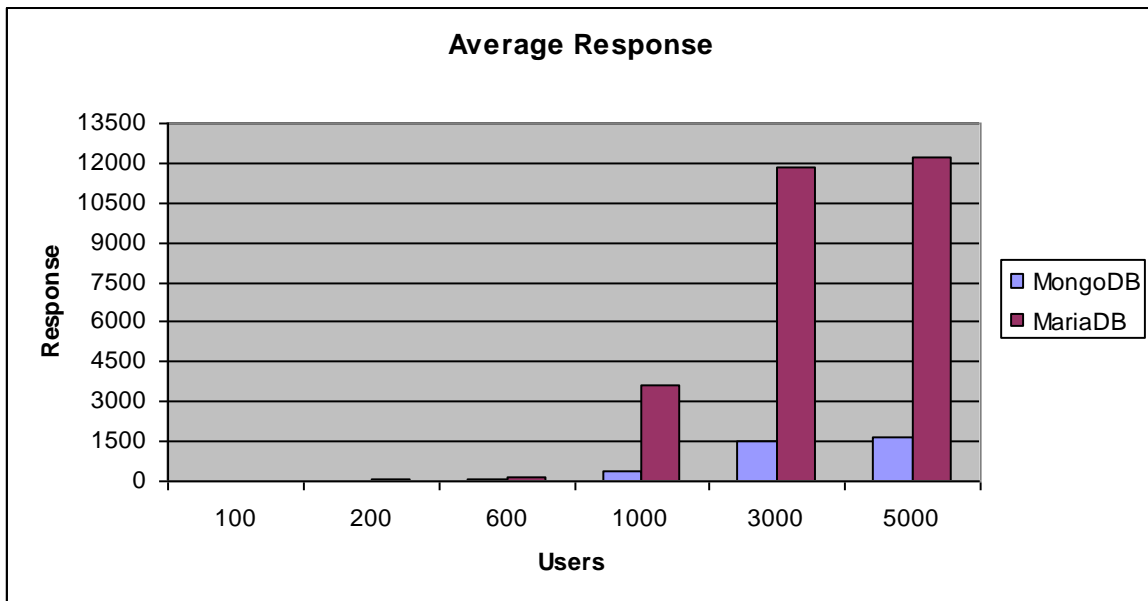


Figure 5.17: Select Query 2 – Avg Responses

5.3.2.7 KB per second for Select query 2 performed on MariaDB

The table below highlights the difference in the KB per second for the MongoDB database when selecting all of the data from two tables and selecting a subset of data from one table.

	100	200	600	1000	3000	5000
Select ALL	666.7	1322.1	3766.8	4663.9	5109	6818.2
Select ALL where Gender = M	0.63	1.26	3.68	6.05	9.29	10.75

Figure 5.18: Select Query 2 – KB per Sec

The graph below illustrates the data that is displayed in the table, figure 5.12 above. This graph highlights the disadvantage of selecting data from more than one table. A MariaDB database encounters more difficulty when working with multiple tables when compared to a MongoDB database.

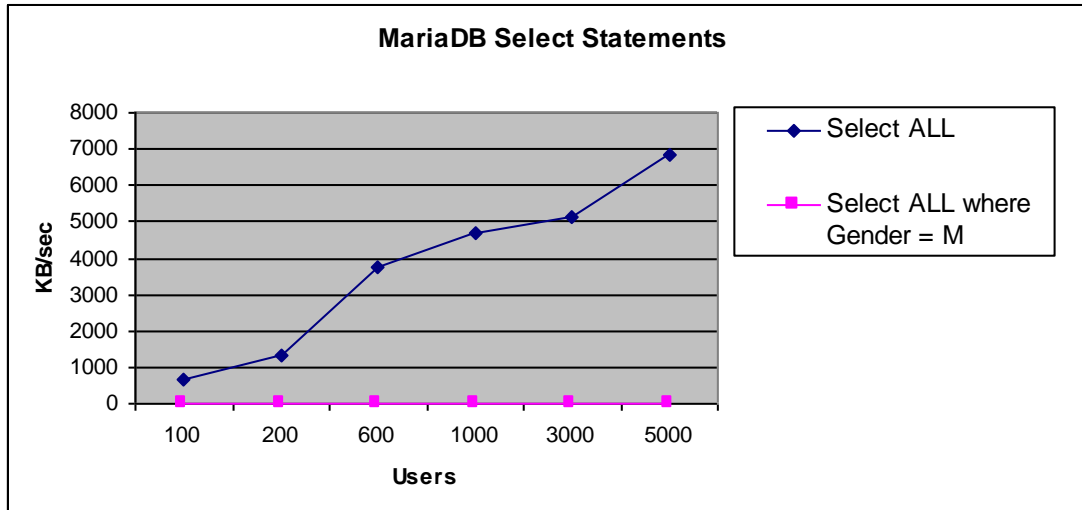


Figure 5.19: Select Query 2 – MariaDB Select Statements

5.3.3 Application Security

To order ensure that applications are not vulnerable or exposed to attacks; developers should follow the OWASP (Open Web Application Security Project) Guidelines. This document details security practices that should be adhere to when developing a web application. As the application developed for this thesis was not a web application not all of the guidelines were applicable. Those that were, were implemented as follows.

Error handling and logging

Error handling and logging is one of the OWASP guidelines that was followed when developing the application. One of the main security concerns to address was the prevention outputting significant database detail when logging information. In this application sensitive information would only be logged when the logging level was set to debug. Error handling and logging also has an effect on performance Error checking forces the application to check for certain conditions before carrying out a request, while logging also can cause a delay as the application has some extra work to do before it can carry on. However the reverse can also have a positive effect on performance as error checking can prevent unnecessary calls being made to the database. It is better practice to check the condition before sending a request to a database as this can save the database from unnecessary processing.

Database security

A connection string should not be hard coded within the application. The connection details for the MariaDB database were stored in a separate database configuration file, which was called by the application. In a larger application this would be done using spring configuration. SQL injection was also prevented. See section 5.3.4 below for more detail.

The connection pool was also closed soon as possible through the application.

Memory management

Similar to the database security, the connection objects and file handlers resources were closed instead of relying on garbage collection.

General coding practices

General coding practices were used through the project, variables and other data sources were explicitly initialised, either during declaration or just before the first usage.

5.4 Security

When developing the application, security was taken into account. Security is an integral part of Big Data applications. Sir Francis Bacon first used the quote “Knowledge is power” in 1597 and this quote is still valid in the modern world of Big Data in relation to the possession and processing of large datasets. A single data type on its own can be insignificant, however when coupled with a second dataset, this data becomes more informative and therefore more valuable. Thus the security of handling this information is paramount.

Ethics has an important role to play when handling personal data of real people. For the purpose of this thesis using live data was not an option as it would require the permission of each passenger.

5.4.1 SQL Injection prevention

SQL Injection attacks are a major concern when developing database applications. SQL injection was prevented in this application by using dropdown boxes to limit the users input.

This ensured that the only data that could be sent to the database from the GUI was that data which was populated in the drop down boxes. This ensured that the search criteria was constructed from the list of pre-defined elements from the drop-down menus. The code for constructing the SQL query for the MariaDB database was constructed using a 'string builder'. This is not in line with OWASP guidelines, however due to time constraints this was the quickest method of querying the database and returning a results set. A version of the application was created using parameterised SQL query with bound parameters. However this made the query difficult to construct due to the nature of the application and the ability of the user to select different combinations of fields.

5.4.2 Data Security

The nature of the aviation industry is that companies often have access to the personal data of passengers travelling on flights. There are types of data that are mandatory for passengers to disclose when travelling, such as name, date of birth, passport number. However, a passenger also has the right to privacy for certain data types.

Companies involved in the aviation industry are in possession large volumes of passenger data. Deriving inferential data is a security issue that can arise when processing passenger and flight data. Inferred data is data about a user that is derived from other data. An example of this in the scenario of passengers and flight pattern is that it would be possible to determine user's nationality based on their departure and return arrival airport. Following on from this it the ability to determine a passengers religion, based on their destination at certain times of the year.

If airlines are going use inferential data for marketing purposes, they need to be careful how they store this data, and they also need to ensure that their derivations are correct. When deriving data types such as passenger's nationality and religious beliefs, companies need to ensure that they are not using this data to discriminate passengers based on their personal information.

Data protection and the security of information are very important to all global aviation organisations and airlines that have access to a customer's personal data. The passenger data that airlines and companies such as SITA have access to, contain a large amount of personal data. To combat this, masking of credit card numbers is a common form of prevention for PCI compliance. In the aviation industry passenger data is the main source of possible breaches. For the purpose of this research it was not possible to use actual passenger data as API data belonged to the airline and the passenger details on those API's belonged to each individual passenger, as a result it would have been necessary to gain permission from each passenger if their data was going to be used.

5.5 Findings

5.5.1 Hypothesis : MongoDB and MariaDB are equally viable options for storing and processing 'Big'.

The initial performance testing outlined above, involved using the application to connect to each database. This test showed that there was little difference in the capability of each database to retrieve and display records. Neither database was dramatically faster or slower than the other. Storing the exact same data in a similar fashion and returning datasets does not yield any considerable discrepancies. It can be noted however that when starting the application while connected to MongoDB, there was a longer delay in the applications drop down boxes being populated.

Both databases offer different capabilities. A MariaDB database can have many tables and allows for the use of 'Joins' to combine data from multiple tables. Contrary to this a MongoDB database allows for data to be stored in one large collection without the restriction of having a. A MongoDB database like MongoDB also offers file storage and mapReduce functionality.

For the purpose of this research a large dataset has been classed as data for 385 flights and 1561 passengers. The size of the data set used was for the MongoDB was 1004KB and for the MariaDB the combined size of the dataset was 352KB. The results that are outlined in

section 4.5 above detail how both databases were able to process the datasets that they contained and under immense loads, using 5000 users.

There are clear differences in performance between MariaDB and MongoDB databases. Section 5.3.1 above highlights the difference in the performance of each database under a load test. Figure 5.9 illustrates that the MongoDB databases outperformed the MariaDB Database. The MongoDB database was able to process a greater number of request per second when under a load.

The speed of processing is an important factor for data that needs to return results quickly. This is particularly true for predictive analysis where real-time data streams are being analysed. In relation to the aviation industry predictive analysis is becoming more prevalent, airlines and airports are interested in knowing in advance when flights will arrive or be delayed and also how quickly the aircraft can turnaround. Calculating this will require many factors such as the weather, the aircrafts fuel, number of passengers on board and others. When predicting flight delays there is a necessity for data to be analysed and results returned in a timely manner.

Figure 5.1 illustrates the number of queries per second for each database; it demonstrates that the MongoDB database be capable of processing more queries than the MariaDB database. Figure 5.7 also demonstrates that the MongoDB database is superior, by presenting the number of average responses for each database. This highlights that the MongoDB database considerably exceeds that of the MariaDB database when inserting data.

Database performance is an important factor in any application; poor performance can impact on the speed at which results can be returned. As mentioned above the ability to produce accurate and usable statistics swiftly is important. The performance of a database is something that has to be considered when selecting a database for a big data project. Creating a schema design will alleviate some issues with database performance. Consequently it is important that time and effort is invested in creating a schema design that will work for the project and will also make the best use of the databases capabilities.

5.6 Evaluation

The testing of the MongoDB database involved more trial and error and the first sample of the application connecting to MongoDB involved trying to join two MongoDB collections, but as Mongo doesn't support joins this was not an easy task. This led to doing a Map Reduce on the two collections (See Appendix A) to combine the results into one table and then running a query on the results of the Map Reduce. The result of this work raises the question about whether using MongoDB is necessary for all types' application or for all data sets. The application had attempted to use MongoDB in a way for which it was not intended to be used. It was through this course of investigation that changed the structure of the data that the application was accessing in MongoDB. The two MongoDB collections (which were based on the relational database model) were combined into one collection.

For the aviation industry using the data from flights and passengers, provides a marketing opportunity for airlines and other businesses. The data that was used in this thesis, will allow the airline to predict trends such as the majority of passengers from a country on a particular route at a certain time of year, or the gender or average of passengers to particular destinations.

6 Conclusions and Future Research

The objective of this thesis was to examine the feasibility of companies in the aviation industry exploring the data that they possess and using it for marketing purposes. Databases were also assessed to investigate what type of database was appropriate to the aviation industry.

The aviation industry has a wealth of data at its disposal. However the harvesting of this data can prove problematic as the data often exists in different locations, making correlation difficult. Aviation industry data is of a structured format and it can be stored in a traditional relational database model.

This thesis demonstrates that for a company to implement a 'Big Data' project, it is not necessary to restructure a system and implement Hadoop. Both MariaDB and MongoDB databases offer practicable options and are viable alternatives to Hadoop in terms of storing and processing data.

The majority of companies implementing big data systems use Hadoop to help with the extracting and analysing of their data. Others are switching to MongoDB databases such as MariaDB which as a database offers more than just data storage capabilities. The Aviation industry does not need to migrate its applications to either of these; it can manage its data using a MariaDB database. The advantage of this is that there is no learning involved and the data is stored as it currently is. However as the testing has shown above, the use of a MongoDB database would improve processing of the data.

The type of data that is available in the aviation industry ranges from aircraft parts to passenger's meals. The issue for most companies is what to do with the data once they have retrieved and stored it. The analysis of the data is a major factor in determining the value of the data and what can be done with it. This thesis has worked with passenger and flight data, which when combined can reveal results pertaining to the age, gender and nationality

of passengers on flight to a particular country. This information becomes valuable once it can be shown to highlight trends in passenger movements.

The main advantage of using a MariaDB database such as MariaDB is that there is minimal set-up involved and for most developers there will be a minimum amount of learning involved. Using a MongoDB database such as MongoDB also required little set-up, however, for most there will be some learning involved in connecting to the database from an application and also in understanding the structure and nature of the data that can be stored.

During the research of this project, the author undertook to set up a MongoDB database (MongoDB) and connect to it from the Java application, which was specifically created to test the data in the database. Setting up MongoDB involved downloading the MongoDB zip file and extracting the files. For a developer with little experience of NoSQL databases, it was challenging to implement the MongoDB database. It was difficult to eliminate the notation of tables, columns and other typical RDBMS functionalities such as primary/foreign keys and Joins.

During the processing and development of the application, a decision was made as to the structure of the data in the MongoDB collection. Following a traditional MariaDB approach lead to having two collections, flights and passengers. However after some struggle to combine the data from the collections it was deemed to be too time consuming for this thesis.

For this research, the data was combined into one large collection to correspond with the principle of a MongoDB database. This poses the question, is MongoDB and Big Data processing the correct format for the Aviation industry, whose data is quite fixed and can be easily formatted to SQL tables? The testing outlined above answers this question. Large data sets that are structured can easily be analysed, and data can be extracted from them in a timely manner. Similarly the data can also be imported into a MongoDB database, however this may add overhead to the project. The author believes that the type of database used would depend on the project and the data that is being used. If it is feasible to move to a

MongoDB database such as MongoDB, then it would undeniably be an advantage, given mapReduce functionality.

The reason that MongoDB and Hadoop have been so influential in the processing of big Data sets is due to the fact that the data to be processed doesn't have to conform to the traditional table formats. MongoDB takes a 'collection' of data, which doesn't have to be formatted in anyway. This allows for different values to be present or not. The data in a MongoDB database, doesn't have to be a structured as an SQL table, for example, a collection can be created that contain the following two documents:

```
{ "_id" : ObjectId("4c2209f9f3924d31102bd84a"), "name" : "mongo" }  
{ "_id" : ObjectId("4c2209fef3924d31102bd84b"), "x" : 3 }
```

Figure 6.1: MongoDB collection example.

Currently big data projects are seen as containing data sets related to social media and online activity, the author believes that this is because it is easier to explain big data when you use an example of random data. This thesis has shown that a big data project can use structured data sets, that may have little insight or information on their own, but when combined with another data set patterns emerge and valuable data information can be uncovered.

In the case of this thesis, the MariaDB database format was easy to apply and the data could be easily searched and the results displayed quite effortlessly. For MongoDB the data had to first be formatted into a collection that MongoDB could process, and then the code of application had to specifically be tailored to suit the results returned. This process was incorrect and it was trying to make the data fit just so MongoDB could be used.

Of the two databases that were chosen for this research, the MongoDB database, MongoDB has proven to be a superior database to use for a Big Data projects. The tests outlined in chapter 5 above highlights the advantages of using a MongoDB database, in the case of MongoDB.

On retrieval of data, the next action for companies is how to analyse and display the results gathered. Charting the results of the data analysis was outside the scope of this thesis. There are an abundance of tools available that can assist such as StatSoft¹³. They have been developing analytical software since 1984 (Statsoft.com, 2014) and have many software solutions for big data, predictive analysis, statistical analysis, data mining and others. There are also many other tools available some of which can be found on the Predictive Analytics Today website¹⁴.

Data visualisation is an important aspect of big data (DataWatch, 2014), as it strives to convey the results of the data analysis. There are an increasing variety of ways to display data rather than the traditional graphs and pie-charts. The use of maps is a modern day trend for representing data. One of the most recognisable mind maps is the Web Trend Map 2007 (See Appendix H). This map uses the Tokyo underground map to display the 200 most successful websites and how they are connected (la.net, 2014).

One of the challenges of data visualisation is identifying how to display the information gathered. There are considerations to take into account before displaying data. It is important to understand the data and to determine what information needs to be visualised. It is also prudent to understand the audience and to convey the information in the most suitable and simplest form. Graph theory tools are available which can facilitate the presentation of statistics by analysing and visualising the data. The investigation of graphing results was outside the scope of this thesis.

Graph tools present many different methods for representing statistics; they also contain algorithms that can be used on the data to reveal patterns. The aviation industry can benefit by using graph tools to generate different visual displays of data, for example using a map of the world is an ideal representation of an airlines flights over a period of time. This type of

¹³ <http://www.statsoft.com/>

¹⁴ <http://www.predictiveanalyticstoday.com/top-10-data-analysis-software/>

visualisation is more meaningful than a bar chart, patterns and trends can be seen much quicker.

Aviation datasets are extremely large. The passenger traffic through Hartsfield-Jackson airport in Atlanta, which was the busiest airport in the world in 2013, was 94.4 million passengers (Burns, 2014). London Heathrow, the busiest airport in Europe handled 72.3 million passengers. In Ireland, 20.2 passengers passed through Dublin airport, (DAA, 2014) and 1.4 million passengers passed through Shannon airport (SAA, 2014). These figures serve as an estimate to the size of datasets that can be expected in the aviation industry. In light of these figures, the author concludes that manual cleaning of datasets in the aviation industry would not be practical. It would involve a large amount of time and effort and could potentially lead to inaccuracies in the data. Therefore tools such as OpenRefine¹⁵ and DataWrangler¹⁶ should be utilized for data scrubbing.

Table 2.1 outlined the ‘Four V’s’ that are often associated with big data. Based on this research, it is the author’s opinion, that they can be prioritised in the following order:

Characteristic	Description	Priority
Veracity	The uncertainty of the data	1
Variety	The different types of data that can exist	2
Velocity	The rate at which the data is being generated	3
Volume	The size of the data being analysed	4

The author believes that veracity is the most important of the four characteristics. Due to the fact that the use of data that is incorrect or out-of-date can result in artificial findings. With passenger data this can be minor discrepancies, such as a person’s status being

¹⁵ <http://openrefine.org/>

¹⁶ <http://vis.stanford.edu/wrangler/>

incorrect, i.e. a woman's status could change from Miss to Mrs. A more detrimental mistake would be to miscalculate a passenger's age. As outlined earlier in section 2.1, the cost of bad data is colossal. Using incorrect data can lead to time wastage and financial loss, therefore there is a necessity to ensure that data accurate.

The variety of the data is the second most important characteristic, as the combination of different datasets can reveal unapparent relationships between the datasets. The amalgamation of unrelated datasets can reveal patterns and trends that would otherwise not be obvious. The data used for the testing in section 4, contains passenger details. If this dataset was combined with a historical data set of passenger details, it would be possible to track a person's airline journey over a period of time, from there it would be possible to track the different planes that were used and the number of times a passenger has been on a particular plane. This is information that cannot be derived by a single set of passenger API data.

The importance of velocity is dependent the purpose of the results of the data. If there is a requirement for real-time stats or live updates to be made to the data, then the velocity at which the data is received is paramount to the big data project. In the aviation industry live data is being processed daily all across the world. The ability to analyse live data is and utilize it is something that airlines and aviation companies should be striving to achieve. A small scale example of this is an airlines ability to send passenger details from a flights departure airport to its destination, before the flight arrives. This allows the airport staff time to receive and analyse the data. The advantage of this is that border control is aware of all passengers on the flight and can identify any potential security threats.

The volume of data can be considered important in view of the fact that the more data that is available, the more accurate the generated results and statistics will be. The size of the dataset used in this thesis was significantly smaller in relation to real world datasets. This suggests that the quality and diversity of data is a higher priority when analysing big data sets. The term 'Big Data' can be misleading as it indicates that a dataset has to be of a considerable size to warrant consideration for analysis. It is the authors opinion that the quality of the data is more important than the quantity.

The limitations of this study were mostly due to time constraints. Big Data is subject that is evolving and new methods and theories are being investigated continuously. To properly test a big data project would require and access to greater infrastructure that was possible for this study. Creating an application that could make use of the capabilities and features of Hadoop is a consideration for further research. Given the time frame and the lack of facilities, it was not possible to set this up. Hadoop does not run on a windows operating system, and installing a VM was not a workable solution.

Limitations on data privacy made obtaining passenger data difficult. After consultation with the legal department of SITA it was decided that it would not be appropriate to use actual passenger API's as under the Data Protection Act it would require the permission of the passengers.

As outlined in 2.3., the aviation industry consists of a large and varied collection of data. The ability to combine all of this data and generate different statistics is a valuable asset for any airline or aviation company. Airports in particular can benefit greatly from predictive analysis. The ability of an airport to predict delays, prevent queues forming and making a passengers experience seamless is invaluable.

7 References

Airline Leader, (2012). How aviation has become a rust belt industry...while squandering a goldmine of unused data. *Airline Leader*, [online] (14), pp.15-31. Available at: <http://www.airlineleader.com/Airline-Leader-Issue-14-Online> [Accessed 18 May. 2014].

Almir, (2013). ISO 27001 Certification. [online] Available at: <http://www.almir.biz/quality-management-systems/information-security-iso-27001/> [Accessed: 8 Dec 2013].

Ayhan, S. Pesce, J. Comitz, P. Sweet, D. Bliesner, S. Gerberick, G. (2013) "Predictive analytics with aviation big data," *Integrated Communications, Navigation and Surveillance Conference (ICNS), 2013*, vol., no., pp.1,13, 22-25 April 2013

Boicea, A, Radulescu, F, Agapin, L I. (2012). MongoDB vs Oracle - database comparison. 2012 Third International Conference on Emerging Intelligent Data and Web Technologies. 1, p 1-3.

Burns, J. (2014). *TOP 10 AIRPORTS: Passenger traffic in 2013*. [online] Airport-world.com. Available at: <http://www.airport-world.com/home/general-news/item/3674-top-10-airports-passenger-traffic-in-2013> [Accessed 24 May. 2014].

CAPA-SITA, (2013). *Transforming the passenger experience Information Technology in Indian Aviation in 2015*. p.10.

Certification Europe, (2013). ISO 27001 Information Security Certification. [online] Available at: <http://certificationeurope.com/iso-27001-information-security/> [Accessed: 8 Dec 2013].

Chaffey, D. (2013). What can established businesses learn from Growth Hacking?. Available: <http://www.smartinsights.com/managing-digital-marketing/marketing-innovation/learn-growth-hacking/>. Last accessed 2nd Dec 2013.

Chakravarty, A. (2012). Big Data – Hadoop from an Infrastructure Perspective. Available: <http://blogs.cisco.com/datacenter/big-data-hadoop-from-an-infrastructure-perspective/>. Last accessed 04th Dec 2013.

Cisco, (2010). Introduction to Context-Aware APIs. In *Cisco Context-Aware API Getting Started Guide*. Cisco Systems, Inc.

Crunchbase.com, (2010). *masFlight|CrunchBase*. [online] Available at: <http://www.crunchbase.com/organization/masflight> [Accessed 18 May. 2014].

DAA, (2014). *Latest News > Passengers at Dublin Airport up 6% to 20.2M in 2013*. [online] Dublinairport.com. Available at: http://www.dublinairport.com/gns/at-the-airport/latest-news/14-01-13/Passengers_at_Dublin_Airport_up_6_to_20_2M_in_2013.aspx [Accessed 24 May. 2014].

Das. T, and Kumar, P. (2013). BIG Data Analytics: A Framework for Unstructured Data Analysis. *International Journal of Engineering and Technology (IJET)*. 5 (1), 153-156

Dataprotection.ie. (2013). Home - Data Protection Commissioner - Ireland. [online] Available at: <https://www.dataprotection.ie/viewdoc.asp?DocID=4> [Accessed: 8 Dec 2013].

DataWatch, (2014). *Fostering Better Decisions With Big Data Visualization*. Available at: http://www.ciosummits.com/Datawatch_Comp_InfoWorld_FosteringBetterDecisionswBigDataVisualization_WP.pdf [Accessed 24 May. 2014].

Docs.mongodb.org, (2013). *Aggregation — MongoDB Manual 2.6.1*. [online] Available at: <http://docs.mongodb.org/manual/aggregation/> [Accessed 16 Nov. 2013].

Erhard, R. and Hong, H. (2014). *Data Cleaning: Problems and Current Approaches*. Leipzig: University of Leipzig.

Gonzalez,H, Halevy,A, Jensen, C S., Langen, A, Madhavan,J, Shapley,R, Shen, W,. (2010). Google Fusion Tables: Data Management, Integration and Collaboration in the Cloud. SoCC '10 Proceedings of the 1st ACM symposium on Cloud computing. p 175-180

Hadoop. (2012). What Is Apache Hadoop?. Available: <http://hadoop.apache.org/>. Last accessed 04th Dec 2013.

Halevy, A., Shapley, R. (2009). Google Fusion Tables. [Blog] *Research Blog: The latest news from Research at Google*. Available at: <http://googleresearch.blogspot.ie/2009/06/google-fusion-tables.html> [Accessed 18 Nov. 2013].

Hutchinson, L. (2012). *Web Served, part 4: Get your database on*. [online] Ars Technica. Available at: <http://arstechnica.com/information-technology/2012/12/web-served-part-4-get-your-database-on/> [Accessed 19 March. 2014].

ia.net, (2014). *Web Trend Map 2007 | Information Architects*. [online] Available at: <http://ia.net/blog/webtrends2007/> [Accessed 23 May. 2014].

iaa.ie, (2012). Irish Aviation Authority successfully re-certified to ISO 9001:2008 Quality Management Standard - Irish Aviation Authority. [online] Available at: <https://www.iaa.ie/news.jsp?i=386> [Accessed 20 May. 2014].

Jadeja, Y. Modi, K (2012) "Cloud computing - concepts, architecture and challenges," Computing, Electronics and Electrical Technologies (ICCEET), 2012 International Conference on , vol., no., pp.877,880, 21-22 March 2012

Larsen, T. (2013). Cross-platform aviation analytics using big-data methods. Integrated Communications, Navigation and Surveillance Conference (ICNS), 2013, 1-13.

Manyika. J, Chui. M, Brown. B, Bughin. J, Dobbs. R, Roxburgh. C, Byers. A. H,. (2011). Big data: The next frontier for innovation, competition, and productivity. Available:

http://www.mckinsey.com/insights/business_technology/big_data_the_next_frontier_for_innovation. Last accessed 3rd Oct 2013.

Masflight.com, (2014). *Advantages | masFlight Aviation Operations Big Data Warehouse and Analytics Platform*. [online] Available at: <http://masflight.com/landing/advantages> [Accessed 28th Nov 2013].

Mukherjee, A. Datta, J. Jorapur, R. Singhvi, R. Haloi, S. Akram, W. (2012) "Shared disk big data analytics with Apache Hadoop," *High Performance Computing (HiPC), 2012 19th International Conference on* , vol., no., pp.1,6, 18-22 Dec. 2012

Murphy, B. (2014). *Big Data and High Performance Computing | Data Storage Management*. [online] [Datastoragereport.com](http://datastoragereport.com). Available at: <http://datastoragereport.com/big-data-and-high-performance-computing> [Accessed 16 May. 2014].

Olston, C. Reed, B. Srivastava, U. Kumar, R. and Tomkins, A. (2008) Pig latin: a not-so-foreign language for data processing. In *Proceedings of the 2008 ACM SIGMOD international conference on Management of data (SIGMOD '08)*. ACM, New York, NY, USA, 1099-1110.

Patel, A.B.; Birla, M.; Nair, U., (2012). Addressing big data problem using Hadoop and Map Reduce, Engineering (NUICONE), Nirma University International Conference on , vol., no., pp.1,5, 6-8

Profilesql.com, (2014). *Free MariaDB profiler - Neor Profile SQL*. [online] Available at: <http://www.profilesql.com/> [Accessed 22 May. 2014].

Rabi. P, (2013). Big Data Processing with Hadoop-MapReduce in Cloud Systems. *International Journal of Cloud Computing and Services Science (IJ-CLOSER)*. 2 (1), 16-27.

Rajesh, Papineni., Latha , Madhavi Y., (2013) HADOOP the Ultimate Solution for BIG DATA Problems. *International Journal of Computer Trends and Technology (IJCTT) - volume4, Issue4 –April 2013*

Schroeck, M, Shockley, R, Smart, J, Romero-Morales, D, Tufano, P. (2012). Analytics: The real-world use of big data. Available: <http://www.ibmbigdatahub.com/whitepapers?page=3>. Last accessed 13th Nov 2013.

SAA, (2014). *Traffic Figures For Shannon Airport*. [online] Shannonairport.ie. Available at: <http://www.shannonairport.ie/gns/about-us/traffic-figures.aspx> [Accessed 24 May. 2014].

SITA, (2013). *Airline IT Trends Survey 2013*. p.12. Available at: <http://www.sita.aero/content/airline-it-trends-survey-2013> [Accessed 16 May. 2014].

Sumranwong, D. "An eLearning Model Application for AS9100 Standard," Information Management, Innovation Management and Industrial Engineering (ICIII), 2011 International Conference on, vol.2, no., pp.173,176, 26-27 Nov. 2011

Tibbetts, H. (2011). \$3 Trillion Problem: Three Best Practices for Today's Dirty Data Pandemic. Available: <http://hollistibbetts.sys-con.com/node/1975126>. Last accessed 10/11/2013.

Vaughan-Nichols, S. (2013). *Google quietly dumps Oracle MariaDB for MariaDB | ZDNet*. [online] ZDNet. Available at: <http://www.zdnet.com/google-quietly-dumps-oracle-MariaDB-for-mariadb-7000020670/> [Accessed 11 Mar. 2014].

Wang, Shidong., Fan , Xingli., Yu, Haiyang., (2011) "The evaluation on the efficiency of airports," Transportation, Mechanical, and Electrical Engineering (TMEE), 2011 International Conference on , vol., no., pp.34,37, 16-18 Dec. 2011

Weijia, Xu., Wei, Luo., Woodward, N., (2012), Analysis and Optimization of Data Import with Hadoop, Parallel and Distributed Processing Symposium Workshops & PhD Forum (IPDPSW), 2012 IEEE 26th International , vol., no., pp.1058,1066, 21-25 May 2012

White, T (2012). Hadoop: The Definitive Guide. 2nd ed. Sebastopol, CA95472: O'Reilly Media Inc. p3-6.

Wood, L. (2013). *Customer Centric Strategy | Business News, Articles & Blogs | SunZu @SunZusocial | By @lyndonx.* [online] SunZu. Available at: <http://www.sunzu.com/articles/customer-centric-strategy/> [Accessed 21 Nov. 2013].

Zhu Wei-ping; Li Ming-xin; Chen Huan, "Using MongoDB to implement textbook management system instead of MySQL," Communication Software and Networks (ICCSN), 2011 IEEE 3rd International Conference on, vol., no., pp.303,305, 27-29 May 2011 doi: 10.1109/ICCSN.2011.6013720

Zoratti, I. (2014). Big Data – An Analysis and Overview. [Blog] SkySQL. Available at: <http://www.skysql.com/blogs/ivan-zoratti/big-data-analysis-and-overview> [Accessed 18 May. 2014].

8 Appendices

Appendix A - JavaScript to map and reduce two collections in MongoDB

At the outset of the development of the application for this thesis, a large portion of time was spent in returning data from MongoDB. The reason for this was that the datasets used were firstly imported into the MariaDB database in the format of two tables. This format was then transferred to the MongoDB database; however it proved tricky to combine the data from the two tables. Part of this reason was the notion that it would be similar to MariaDB, however MongoDB does not support 'JOINS', and trying to combine data from two collections goes against the principle idea behind MongoDB and Big Data processing. One of the advantages of a MongoDB database like MongoDB is that there is no need for data to be structured. It is preferable to have all of the data in one collection in any format. Appendix A shows that using JavaScript it is possible to extract data from two collections in MongoDB using mapReduce.

```
var mapFlights, mapPassengers, reduce;

mapFlights = function() {

    emit(this.FlightNumber, {PNR_LOC: this.PNR_LOC,
        ArrivalAirport : this.ArrivalAirport,
        FirstName: this.FirstName,
        SecondName: this.SecondName,
        Age: this.Age,
        Nationality : this.Nationality,
        Gender: this.Gender});

};

mapPassengers = function() {

    emit(this.FlightNumber, {PNR_LOC: this.PNR_LOC,
        ArrivalAirport : this.ArrivalAirport,
```

```

    FirstName: this.FirstName,
    SecondName: this.SecondName,
    Age: this.Age,
    Nationality : this.Nationality,
    Gender: this.Gender});
};

```

```

reduce = function(key, values) {
var result = {
    "PNR_LOC": '',
    "ArrivalAirport": '',
    "FirstName": '',
    "SecondName": '',
    "Age": '',
    "Nationality" : '',
    "Gender": ''
};

```

```

values.forEach(function(value) {
    if (result.PNR_LOC == "" && value.PNR_LOC !== null) {
        result.PNR_LOC = value.PNR_LOC;
    }

    if (result.ArrivalAirport == "" && value.ArrivalAirport !== null) {
        result.ArrivalAirport = value.ArrivalAirport;
    }

    if (result.FirstName == "" && value.FirstName !== null) {
        result.FirstName = value.FirstName;
    }

    if (result.SecondName == "" && value.SecondName !== null) {
        result.SecondName = value.SecondName;
    }

    if (result.Age == "" && value.Age !== null) {
        result.Age = value.Age;
    }

    if (result.Nationality == "" && value.Nationality !== null) {
        result.Nationality = value.Nationality;
    }

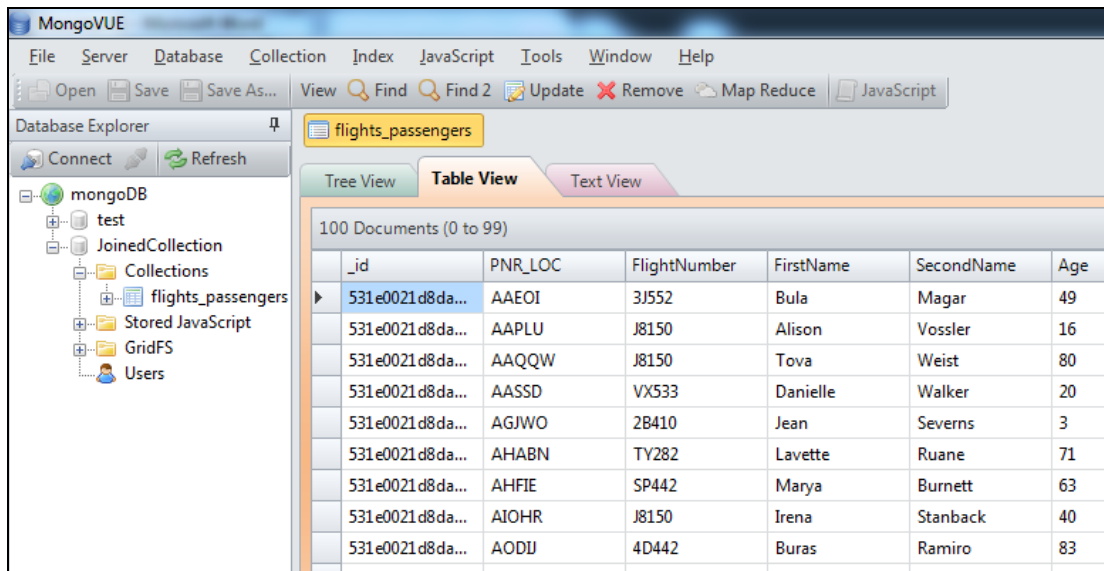
    if (result.Gender == "" && value.Gender !== null) {
        result.Gender = value.Gender;
    }
}

```

```
    }  
  
  });  
  
  return result;  
};  
  
db.passengers.mapReduce(mapPassengers, reduce, {"out": {"reduce":  
"flights_passengers"}});  
db.flights.mapReduce(mapFlights, reduce, {"out": {"reduce":  
"flights_passengers"}});
```

Appendix B

This is a screen shot of the data contained in the MongoDB. A tool called MongoVUE was used to view the data. This is a view of the 'Table View' similar to how the data would look in a RDBMS.



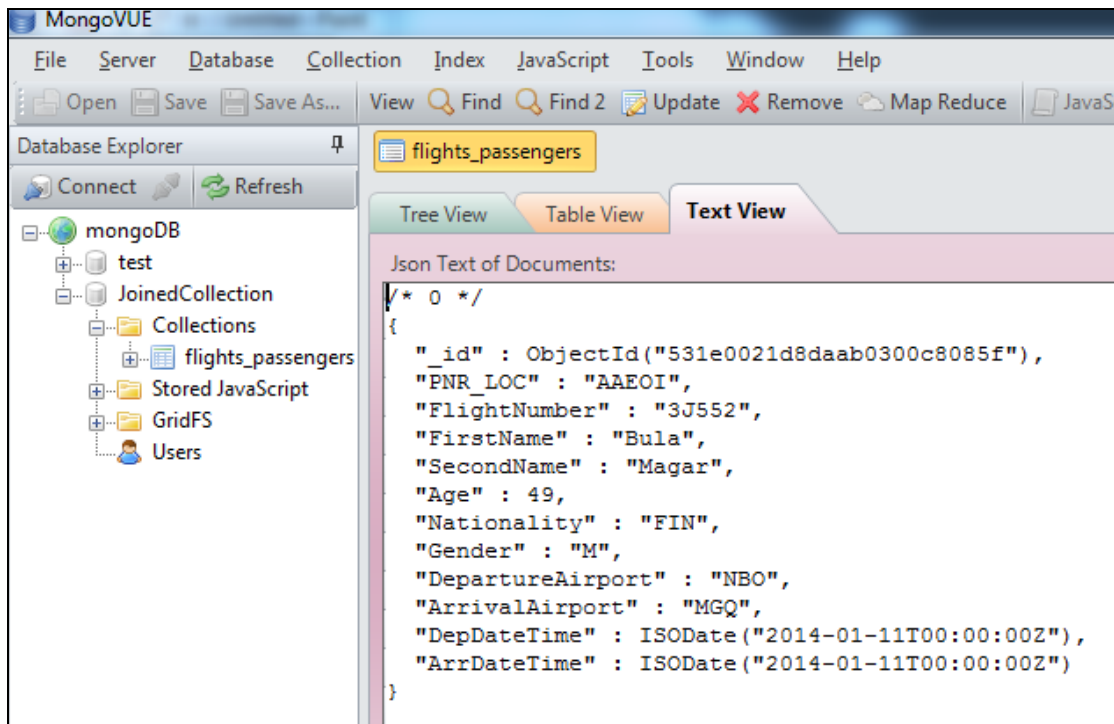
The screenshot shows the MongoVUE application window. The 'Database Explorer' on the left shows a tree view with 'mongoDB' expanded to 'test', which contains 'JoinedCollection', 'Collections', 'flights_passengers', 'Stored JavaScript', 'GridFS', and 'Users'. The main area displays the 'flights_passengers' collection in 'Table View' mode, showing 100 documents (0 to 99). The table has the following columns: _id, PNR_LOC, FlightNumber, FirstName, SecondName, and Age. The first few rows of data are as follows:

_id	PNR_LOC	FlightNumber	FirstName	SecondName	Age
531e0021d8da...	AAEOI	3J552	Bula	Magar	49
531e0021d8da...	AAPLU	J8150	Alison	Vossler	16
531e0021d8da...	AAQQW	J8150	Tova	Weist	80
531e0021d8da...	AASSD	VX533	Danielle	Walker	20
531e0021d8da...	AGJWO	2B410	Jean	Severns	3
531e0021d8da...	AHABN	TY282	Lavette	Ruane	71
531e0021d8da...	AHFIE	SP442	Marya	Burnett	63
531e0021d8da...	AIOHR	J8150	Irena	Stanback	40
531e0021d8da...	AODIJ	4D442	Buras	Ramiro	83

Appendix C

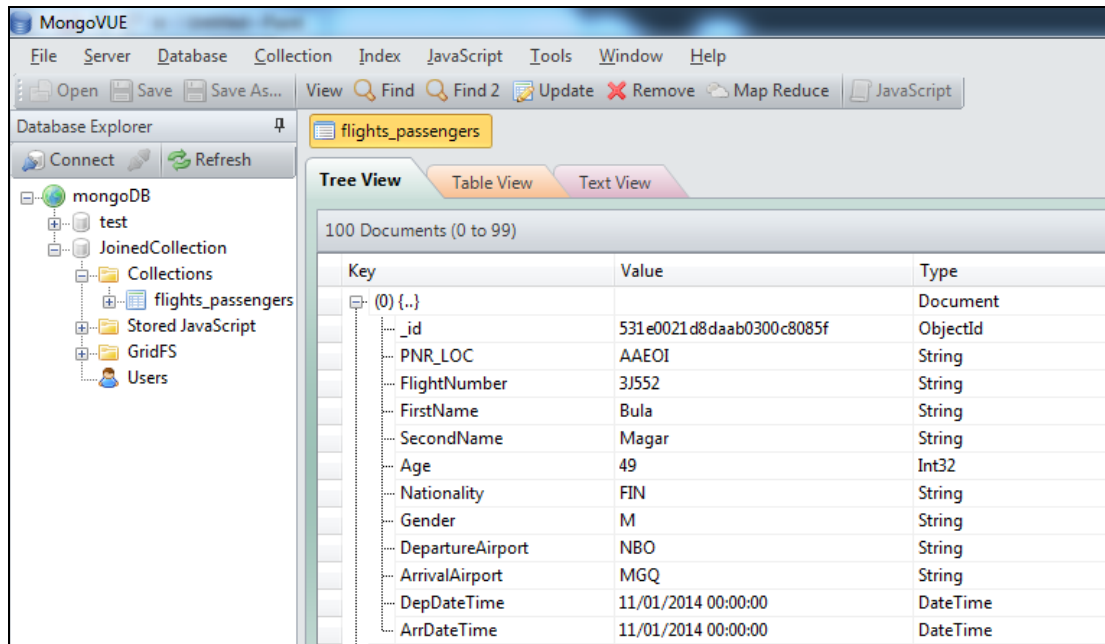
MongoVUE Text View

This is a screen shot of the data contained in the MongoDB. A tool called MongoVUE was used to view the data. This is a view of the 'Text View' which displays the data in JSON format



Appendix D

This is a screen shot of the data contained in the MongoDB. A tool called MongoVUE was used to view the data. This is a view of the 'Tree View' which allows the user to expand out the JSON objects.



The screenshot shows the MongoVUE application interface. The 'Database Explorer' on the left shows a tree structure with 'mongoDB' at the root, containing 'test', 'JoinedCollection', 'Collections', 'Stored JavaScript', 'GridFS', and 'Users'. The 'Collections' folder is expanded to show 'flights_passengers'. The main window displays the 'Tree View' of a document in the 'flights_passengers' collection. The document is expanded to show its fields and values.

Key	Value	Type
(0) {...}		Document
.._id	531e0021d8daab0300c8085f	ObjectId
..PNR_LOC	AAEOI	String
..FlightNumber	3J552	String
..FirstName	Bula	String
..SecondName	Magar	String
..Age	49	Int32
..Nationality	FIN	String
..Gender	M	String
..DepartureAirport	NBO	String
..ArrivalAirport	MGQ	String
..DepDateTime	11/01/2014 00:00:00	DateTime
..ArrDateTime	11/01/2014 00:00:00	DateTime

Appendix E

Test steps to confirm that the application is connecting to the database.

Verify MongoDB connection

- Start the application.
- Select each of the drop down menus, verify that they only contain the value 'ALL'.
- Check the console window/log file for the following error message:

```
May 05, 2014 5:42:32 PM com.mongodb.DBTCPConnector initDirectConnection  
WARNING: Exception executing isMaster command on localhost/127.0.0.1:27017  
java.net.ConnectException: Connection refused: connect
```

- Start up the MongoDB database by running the following command:

```
mongod
```

- Return to the GUI and select each of the drop down menus and verify that all of them (excluding the gender menu) only contain the value 'ALL'
- Check the console window /log file for the successful database connection messages and the name of the collection that the application is connected to.

```
Connection to MongoDB successful  
flights_passengers
```

Verify MariaDB connection

- Remove the database configuration file (db.cfg) from the resources directory
- (..\FlightData\src\main\resources).
- Start the application.
- Select each of the drop down menus and verify that all of them (excluding the gender menu) only contain the value 'ALL'
- Check the console window/log file for the following error message:

```
ERROR connecting to databasejava.io.FileNotFoundException:  
C:\Dev\ThesisPractical_MariaDB\FlightData\src\main\resources\mydb.cfg (The  
system cannot find the file specified)  
  
java.io.FileNotFoundException:  
C:\Dev\ThesisPractical_MariaDB\FlightData\src\main\resources\mydb.cfg (The  
system cannot find the file specified)
```

- Replace the database configuration file and restart the application.
- Select each of the drop down menus, verify that they are populated with data from the database.
- Check the console window /log file for a successful database connection messages.

Connection Successful!org.mariadb.jdbc.MariaDBConnection@6ff03fae

Appendix F

Code

These are held in the Code folder. All instructions for set-up and running of the code are held in a README file.

Appendix G

STANDARD: Potentially Unsafe Code - java.io.File

Line: 3 - C:\Users\Maria\Desktop\VCG_Test\MariaDB\DBConnection.java

This functionality acts as an entry point for external data and the code should be manually checked to ensure the data obtained is correctly validated and/or sanitised. Additionally, careful checks/sanitisation should be applied in any situation where the user may be able to control or affect the filename.

STANDARD: Potentially Unsafe Code – FileInputStream

Line: 4 - C:\Users\Maria\Desktop\VCG_Test\MariaDB\DBConnection.java

This function acts as an entry point for external data and the code should be manually checked to ensure the data obtained is correctly validated and/or sanitised. Additionally, careful checks/sanitisation should be applied in any situation where the user may be able to control or affect the filename.

STANDARD: Potentially Unsafe Code - java.io.File

Line: 4 - C:\Users\Maria\Desktop\VCG_Test\MariaDB\DBConnection.java

This functionality acts as an entry point for external data and the code should be manually checked to ensure the data obtained is correctly validated and/or sanitised. Additionally, careful checks/sanitisation should be applied in any situation where the user may be able to control or affect the filename.

STANDARD: Potentially Unsafe Code – FileInputStream

Line: 39 - C:\Users\Maria\Desktop\VCG_Test\MariaDB\DBConnection.java

This function acts as an entry point for external data and the code should be manually checked to ensure the data obtained is correctly validated and/or sanitised. Additionally, careful checks/sanitisation should be applied in any situation where the user may be able to control or affect the filename.

LOW: Potentially Unsafe Code - Operation on Primitive Data Type

Line: 184 - C:\Users\Maria\Desktop\VCG_Test\MariaDB\DBConnection.java

The code appears to be carrying out a mathematical operation on a primitive data type. In some circumstances this can result in an overflow and unexpected behaviour. Check the code manually to determine the risk.

LOW: Potentially Unsafe Code - Operation on Primitive Data Type

Line: 185 - C:\Users\Maria\Desktop\VCG_Test\MariaDB\DBConnection.java

The code appears to be carrying out a mathematical operation on a primitive data type. In some circumstances this can result in an overflow and unexpected behaviour. Check the code manually to determine the risk.

Appendix H



Appendix I - Code Testing

A tool called Visual Code Grepper (VCG) was used to analyse the code that was used to connect to each database. Using this tool the DBConnection.java file was loaded in for each instance of the application. Figure 5.19X below shows the break down for MariaDB, it showed that there were potential 6 dangerous code violations. However upon investigation, 2 of these error or low priority and were due to operations on primitive data types, the other 4 were warnings based on the file I/O imports that were being used. Appendix G contains the complete list of the errors that were highlighted.

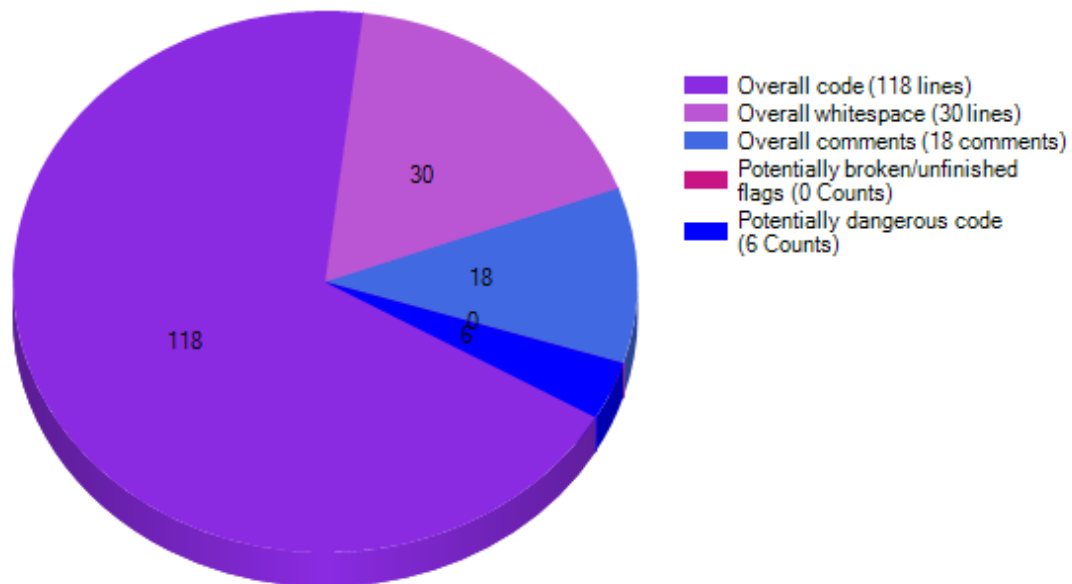


Figure 5.1: MariaDB Code analysis

Figure 5.20 below shows the breakdown of code violations for MongoDB. The database connection details for MongoDB were hard coded into the application and this is the reason why it didn't have the file I/O concerns that MariaDB had.

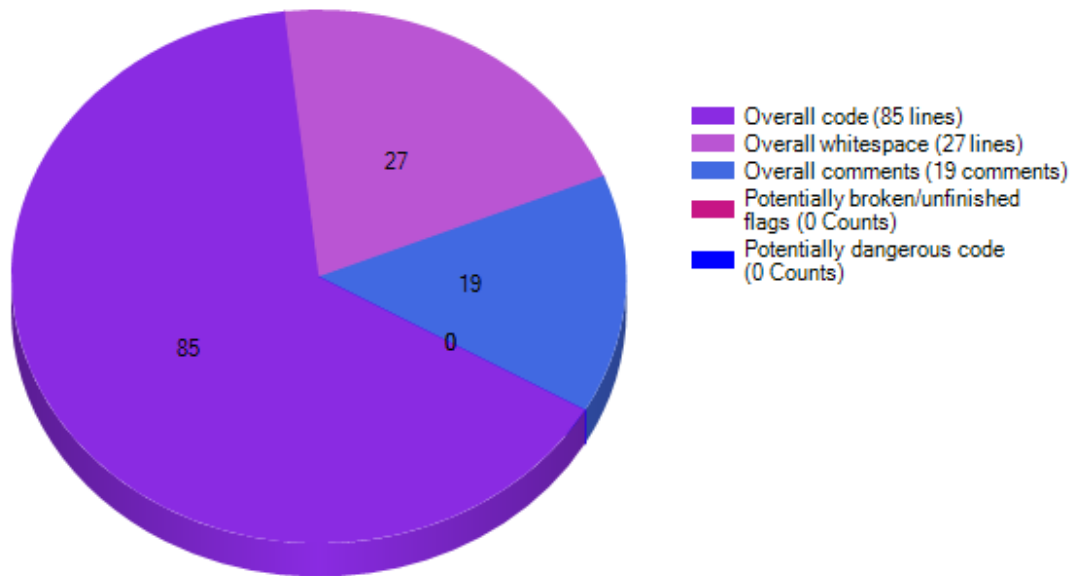


Figure 5.2: MongoDB Code analysis

The graph below shows a comparison of the results for both connections. There were no major security concerns with either database connection. Having more time to work on this, using Spring configuration to load in the database configuration would be ideal and a viable option to avoid the potential security threats posed by using file I/O. It would also be a better way than hard coding it into the application.

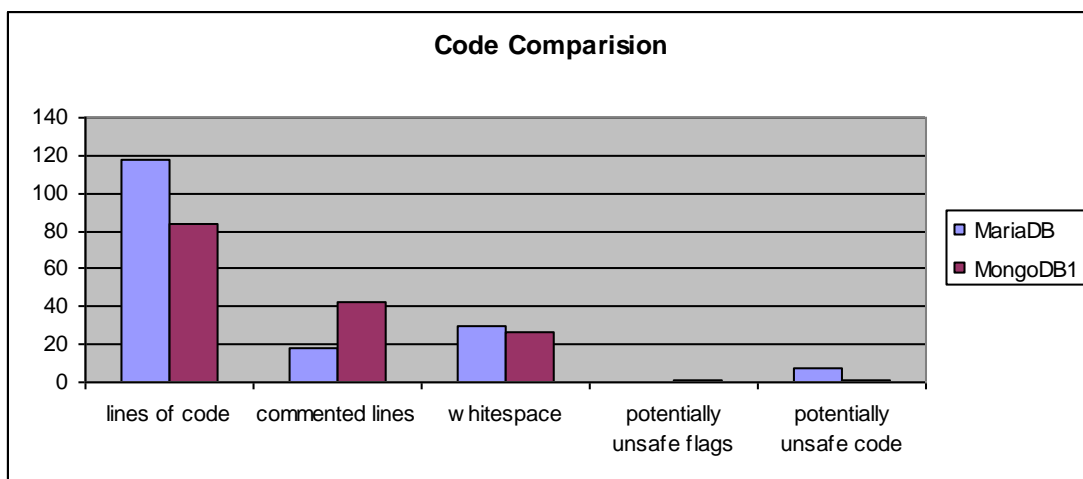


Figure 5.3: MariaDB V Mongo Code