An artefact to analyse unstructured document data stores

by

André Romeo Botes

20953259

Dissertation submitted in fulfilment of the requirements for the degree

MAGISTER SCIENTIA IN COMPUTER SCIENCE

in the
SCHOOL OF INFORMATION TECHNOLOGY

at the
VAAL TRIANGLE CAMPUS

of the
North-West University

Vanderbijlpark

Supervisor: Prof. Roelien Goede

Co-Supervisor: Imelda Smit

2014

(PAGE INTENTIONALLY LEFT BLANK)

DECLARATION

I, André Romeo Botes declare that

An artefact to analyse unstructured document data stores

is my own work and that all the sources I have used or quoted have been indicated and acknowledged by means of complete references.

Signature: 

Date: 6 December 2013

ACKNOWLEDGEMENTS

Embarking on this journey of research leading to the compilation of this dissertation was a process of exploration and discovery. The rewards were fulfilling and fruitful and led me down a path of self-discovery. However, I would like to acknowledge the following people who made this expedition possible:

Firstly, I would like to give special thanks to Prof. Roelien Goede, my supervisor, for her input, guidance and patience.

Secondly, I would like to give special thanks to Mrs Imelda Smit, my co-supervisor, for her support, encouragement and motivation.

Thirdly, I would like to thank Mrs Natasha Ravyse and CTrans, for their input by highlighting many grammatical and technical errors that would have gone unnoticed otherwise.

Finally, I would like to thank my parents, André and Marie Botes, for their support and encouragement.

UNICUIQUE SUUM

ABSTRACT

Structured data stores have been the dominating technologies for the past few decades. Although dominating, structured data stores lack the functionality to handle the ‘Big Data’ phenomenon. A new technology has recently emerged which stores unstructured data and can handle the ‘Big Data’ phenomenon.

This study describes the development of an artefact to aid in the analysis of NoSQL document data stores in terms of relational database model constructs. Design science research (DSR) is the methodology implemented in the study and it is used to assist in the understanding, design and development of the problem, artefact and solution.

This study explores the existing literature on DSR, in addition to structured and unstructured data stores. The literature review formulates the descriptive and prescriptive knowledge used in the development of the artefact. The artefact is developed using a series of six activities derived from two DSR approaches.

The problem domain is derived from the existing literature and a real application environment (RAE). The reviewed literature provided a general problem statement. A representative from NFM (the RAE) is interviewed for a situation analysis providing a specific problem statement.

An objective is formulated for the development of the artefact and suggestions are made to address the problem domain, assisting the artefact’s objective.

The artefact is designed and developed using the descriptive knowledge of structured and unstructured data stores, combined with prescriptive knowledge of algorithms, pseudocode, continuous design and object-oriented

design. The artefact evolves through multiple design cycles into a final product that analyses document data stores in terms of relational database model constructs.

The artefact is evaluated for acceptability and utility. This provides credibility and rigour to the research in the DSR paradigm. Acceptability is demonstrated through simulation and the utility is evaluated using a real application environment (RAE). A representative from NFM is interviewed for the evaluation of the artefact.

Finally, the study is communicated by describing its findings, summarising the artefact and looking into future possibilities for research and application.

Keywords: design science research, structured data stores, unstructured data stores, artefact, NoSQL, big data.

UITTREKSEL

Gestruktureerde datastore was die afgelope paar dekades die oorheersende tegnologie. Alhoewel dit die geval was, het dit funksionele tekortkominge gehad ten opsigte van die hantering van die “Big Data”-verskynsel. Onlangs het ’n nuwe tegnologie na vore gekom wat ongestruktureerde data stoor en die “Big Data”-verskynsel beter kan ondersteun.

Hierdie studie beskryf die ontwikkeling van ’n artefak om die ontleding van die NoSQL-dokumentdatastore te vergemaklik deur gebruik te maak van die relasionele databasisstrukture. Die ontwerp- wetenskaplike navorsingsmetode (OWN) word in hierdie studie gebruik om die begrip, ontwerp en ontwikkeling van die probleem, artefak en oplossing te fasiliteer.

Die studie ondersoek bestaande literatuur gebaseer op OWN saam met literatuur oor gestruktureerde en ongestruktureerde datastore. Die hersiende literatuur formuleer die beskrywende en voorskriftelike kennis wat in die ontwikkeling van die artefak gebruik word. Die artefak is ontwikkel met die behulp van ’n reeks van ses aktiwiteite wat spruit uit twee OWN benaderings.

Die probleemdomein word afgelei vanuit die bestaande literatuur en vanuit ’n werklike toepassingomgewing (WTO). Die literatuur voorsien ’n algemene probleemverklaring. ’n Onderhoud word gevoer met die verteenwoordiger van NFM (die WTO) vir ’n situasie-analise vir die spesifieke probleemstellingverklaring.

’n Doelwit is geformuleer vir die ontwikkeling van die artefak en voorstelle word gemaak om die probleem op te los wat die artefak se doelwit ondersteun.

Die artefak word ontwerp en ontwikkel met behulp van die beskrywende kennis van gestruktureerde en ongestruktureerde datastore wat gekombineer word met voorskriftelike kennis van algoritmes, pseudokode, voortdurende ontwerp en objekgeoriënteerde ontwerp. Die artefak word verfyn deur gebruik te maak van verskeie ontwerpsiklusse. ’n Finale artefak word ontwikkel wat dokumentdatastore in terme van relasionele databasismodelkonstrukte ontleed.

Die artefak word geëvalueer vir aanvaarbaarheid en bruikbaarheid. Dit lewer geloofwaardige en deeglike navorsing in die OWN-paradigma. Aanvaarbaarheid word gedemonstreer deur simulasie, en bruikbaarheid word geëvalueer met behulp van ’n werklike toepassingomgewing (WTO). ’n Onderhoud word gevoer met die verteenwoordiger van NFM vir die evaluering van die artefak.

Laastens word die studie gekommunikeer deur die beskrywing van bevindinge, ’n opsomming van die artefak en ’n ondersoek na toekomstige moontlikhede vir navorsing en aanwending.

Sleutelwoorde: ontwerp- wetenskaplike navorsing, gestruktureerde datastore, ongestruktureerde datastore, artefak, NoSQL, “Big Data”.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

LIST OF CODE SEGMENTS

CHAPTER ONE: INTRODUCTION

1.1 INTRODUCTION

In recent years, a new attitude towards data has emerged: data that have always been in existence, and in many cases, could have been developed as a strategic commodity in organisations have now been discovered. These data have been accumulated by companies over the years, lying dormant, only to be valued now. The afore-mentioned data contain hidden information that could assist organisations with information that may be crucial for decision making. The data have characteristics such as being high volume, high frequency, and mostly semi-structured. The data are retrieved from multiple and varied sources and need to be managed and used to the organisation's benefit. This perspective on data triggered a phenomenon called 'Big Data'. According to Morris *et al.* (2013:88):

> *"Big Data refers to a movement to find new and better ways to manage large amounts of web-generated data, derive business insight from it, while also providing high performance and scalability at a reasonable cost".*

The large volumes of web-generated data were the main motivation for the development of unstructured data stores. Morris *et al.* (2013:10) define unstructured data as "data that exist in their original (raw) state; that is in the format in which they were collected", whereas structured data are defined as "unstructured data that have been formatted (structured) to facilitate storage, use, and information generation".

This study aims to design and develop an artefact to analyse unstructured document data stores in terms of structured data model constructs. This artefact is aimed at assisting in the analysis of unstructured document data stores in order to present formatted and interpretable output of the data for information generation.

The objective of this chapter is to orientate the study by introducing the motivation for this study (1.2); aspects central to the study (1.3); the research objectives (1.4); the research methodology employed in this study (1.5); chapter classification (1.6); and finally, the conclusion (1.7).

1.2 MOTIVATION FOR THIS STUDY

Newcom Fluid Management Pty (NFM) is a business that has been producing big data. Over a number of years, NFM has monitored fluid usage and consumption in multiple industries. NFM has implemented a number of monitoring devices without realising the full potential of the data these devices could provide. The data include stored accounts of the fluids dispensed. Examples of known structured data used in NFM include consumer identifiers, user identifiers, odometer readings, volume expelled and timestamps. The business representative, Mr Francois Oosthuizen, suspects that more data are produced by the monitoring devices than are utilised by the business because of the unstructured origin of the data (Oosthuizen, 2013a). Design science research (DSR) offers an opportunity to develop intuitions into feasible ideas. This is supported by Hevner *et al.* (2004:99), who state that the existing knowledge base is often insufficient for design purposes and designers must rely on intuition, experience and trial-and-error methods.

The dilemma that an organisation such as NFM is confronted with, is: How can more information be extracted from the data produced by the monitoring devices?

A problem most organisations are confronted with is that they cannot efficiently utilise the information represented by unstructured data. These unstructured data are not easily integrated with structured data to allow optimal utilisation in organisations. Newcom Fluid Management (NFM) is an example of such an organisation. Their monitoring devices are capable of gathering unstructured information. Unstructured data can be captured in the form of extra data packets containing data on device status, tank levels and environmental factors such as humidity and temperature. The data of NFM are presented in Figure 1.1 and Appendix A, illustrating this dilemma in the context of this study and the real application environment (RAE).


{ "_id" : ObjectId("52378add6058a7c73e239fd7"), "record" : "15", "consumer_uid" : "S2102", "customer_uid" : "CUSTOMER103
"consumer_group_uid" : "C2GROUP107", "consumer_type_uid" : "S2TYPE1", "tag" : "22222", "name" : "SIM002GP", "secondary_name" : nul
"description" : "VOLVO LOCAL 2", "current_odometer" : "50000", "prevoius_odometer" : null, "capacity" : "400", "daily_limit" : "100", "lk_cpk" : nul
"current_limit" : null, "current_limit_date" : null, "current_hours" : null, "prevoius_hours" : null, "capture_type" : "O", "km_limit" : "400", "setup"
"KMHR%REASON%VOLUME%WORKER", "min_kpl" : null, "max_kpl" : null, "transaction" : [], "events" : [], "groups" : [{ "record" : "7
"consumer_group_uid" : "C2GROUP107", "customer_uid" : "CUSTOMER103", "name" : "CROSS BORDER", "description" : null, "other" : nul
"min_kpl" : null, "max_kpl" : null }], "types" : [{ "record" : "8", "consumer_type_uid" : "S2TYPE1", "customer_uid" : "CUSTOMER103", "name"
"SMALL EQUIPMENT", "description" : null, "other" : null, "min_kpl" : null, "max_kpl" : null }] }{ "_id" : ObjectId("52378add6058a7c73e239fe6"
"record" : "71", "consumer_uid" : "VEHARBTVB2013032021", "customer_uid" : "ARBTVB20130320", "consumer_group_uid"
"GP1ARBTVB20130321", "consumer_type_uid" : "TP1ARBTVB20130321", "tag" : "307E73089CA9", "name" : "F121", "secondary_name"
"YYC243GP", "description" : "Please enter", "current_odometer" : "132288", "prevoius_odometer" : null, "capacity" : "200", "daily_limit" : "200
"lk_cpk" : "0.00", "current_limit" : null, "current_limit_date" : null, "current_hours" : null, "prevoius_hours" : null, "capture_type" : "O", "km_limit"
"500", "setup" : "KMHR%REASON%VOLUME%WORKER", "min_kpl" : "7", "max_kpl" : "11", "transaction" : [{ "record" : "335
"consumer_transaction_uid" : "48da8547-a818-11e2-b3d0-001c14012e75", "udatetime" : "2013-04-18 01:02:25", "consumer_uid"
"VEHARBTVB2013032021", "worker_uid" : "OP1ARBTVB201303207", "volume" : "0.061", "odometer" : "132302.000", "hours" : "0", "source"
"Internal", "location" : "Unknown", "price" : "0.06", "remarks" : "Other Filling", "start_time" : "2013-04-18 01:01:57", "end_time" : "2013-04-1
01:02:32", "cpk" : "0.00", "kpl" : "0.00", "bypassed" : "NO", "auth_uid" : "Unknown", "restype" : "Normal", "trip_distance" : "0" }, { "record" : "425
"consumer_transaction_uid" : "8a2f250b-ae40-11e2-b3d0-001c14012e75", "udatetime" : "2013-04-26 09:08:17", "consumer_uid"
"VEHARBTVB2013032021", "worker_uid" : "OP1ARBTVB201303223", "volume" : "49.413", "odometer" : "135125.000", "hours" : "0", "source"
"Internal", "location" : "Unknown", "price" : "49.41", "remarks" : "Normal Filling", "start_time" : "2013-04-26 09:07:39", "end_time" : "2013-04-2
09:09:40", "cpk" : "57.13", "kpl" : "57.13", "bypassed" : "NO", "auth_uid" : "Unknown", "restype" : "Normal", "trip_distance" : "2823" }, { "record"
"467", "consumer_transaction_uid" : "6116f40d-b3c1-11e2-b3d0-001c14012e75", "udatetime" : "2013-05-03 09:04:22", "consumer_uid"
"VEHARBTVB2013032021", "worker_uid" : "OP1ARBTVB201303221", "volume" : "48.900", "odometer" : "135524.000", "hours" : "0", "source"
"Internal", "location" : "Unknown", "price" : "48.90", "remarks" : "Normal Filling", "start_time" : "2013-05-03 09:03:04", "end_time" : "2013-05-0
09:05:48", "cpk" : "8.16", "kpl" : "8.16", "bypassed" : "NO", "auth_uid" : "Unknown", "restype" : "Normal", "trip_distance" : "399" }], "events" :
"record" : "289", "data_event_uid" : "1b85b70e-ddd1-4e74-9dc6-eb2015ac36fc", "udatetime" : "2013-05-19 11:55:04", "customer_uid" : "0
"consumer_uid" : "VEHARBTVB2013032021", "depot_uid" : "NA", "station_uid" : "NA", "container_uid" : "NA", "worker_uid" : "0", "event_name"
"Consumer F121 KPL below Minimum", "event_type" : "Consumer KPL Below Minimum", "description" : "Consumer F121 KPL of 0.000000 i
below the minimum of 7.", "processed" : "1" }, { "record" : "535", "data_event_uid" : "52153f07-d291-44e8-b3ff-0d1404f13f25", "udatetime"
"2013-06-02 11:55:00", "customer_uid" : "ARBTVB20130320", "consumer_uid" : "VEHARBTVB2013032021", "depot_uid" : "NA", "station_uid"
"NA", "container_uid" : "NA", "worker_uid" : "0", "event_name" : "Consumer F121 KPL below Minimum", "event_type" : "Consumer KPL Belo
Minimum", "description" : "Consumer F121 KPL of 0.000000 is below the minimum of 7.", "processed" : "1" }], "groups" : [{ "record" : "10
"consumer_group_uid" : "GP1ARBTVB20130321", "customer_uid" : "ARBTVB20130320", "name" : "Bakkies", "description" : null, "other" : nul
"min_kpl" : null, "max_kpl" : null }], "types" : [{ "record" : "12", "consumer_type_uid" : "TP1ARBTVB20130321", "customer_uid"
"ARBTVB20130320", "name" : "Bakkies", "description" : null, "other" : null, "min_kpl" : null, "max_kpl" : null }] }{ "_id"

Figure 1.1: Subset of RAE data

The main motivation for this study is to develop an artefact to address the problem organisations are confronted with when utilising unstructured data. The aim of the artefact is to analyse unstructured data stores, specifically document data stores, to identify information that may be utilised by these organisations.

1.3 ASPECTS CENTRAL TO THIS STUDY

The sections that follow introduce key aspects of the study, namely: design science research, NFM as a client, structured data stores, unstructured data stores and a comparison of structured and unstructured data stores.

1.3.1 Design science research

Design science research (DSR) has been put into practise for some time in the engineering and Information Systems (IS) disciplines (Gregor & Hevner, 2013:338). DSR is defined in the IS discipline as research that is concerned with the

construction of a wide range of socio-technical artefacts such as decision support systems, modelling tools, governance strategies, methods of IS evaluations, and IS change interventions (Gregor & Hevner, 2013:337). DSR also analyses the performance of a designed artefact in order to understand and improve the artefact (Vaishnavi & Kuechler, 2004; Hevner & Chatterjee, 2010:30). DSR is primarily the creation and evaluation of an artefact used to acquire the solution to the identified organisational problem by understanding it (Hevner *et al.*, 2004:82; Hevner & Chatterjee, 2010:6). The evaluation of these artefacts could be subject to quantitative and qualitative empirical methods (Hevner *et al.*, 2004).

Several approaches are available to guide researchers in performing DSR research. The approaches suggested by Peffers *et al.* (2008) and by Vaishnavi and Kuechler (2004) are key research approaches of this study, and a combination of these approaches is presented in Chapter 2.

1.3.2 Newcom Fluid Management

Since the practical application of the artefact in a real-world environment is important in DSR research, a client environment has been identified. Newcom Fluid Management (NFM) is used as the real application environment (RAE) in this DSR study. To understand the need for the artefact, background information on NFM is given.

In June 2011, NFM officially started operations and currently has a wide footprint in the South African market of operational fluid management systems. They serve clients in different market segments, which have resulted in NFM having grown substantially in the short time since its inception.

Various skills, technological and business resources and experience have been acquired by NFM, providing them with the advantage of being able to deliver cost-effective, high-quality fuel management solutions to clients. NFM's goals are to provide the best fuel management solutions for their clients.

The above section addressed the context of NFM. A description of their specific needs with regard to the utilisation of their unstructured data is presented in Chapter 5.

1.3.3 Structured data stores

A database is a construct used to facilitate storage, use and information generation of structured data (formatted unstructured data) (Morris *et al.*, 2013:10).

A database is defined firstly as a collection of raw facts or end-user data, and secondly as metadata or data about data (Morris *et al.*, 2013:7). It typically describes the activities of one or more related organisations as well (Ramakrishnan & Gehrke, 2003:4). Databases evolved alongside database management systems (DBMSs) to enable the effective handling of databases. A DBMS is software designed to assist in maintaining and utilising large collections of data (Ramakrishnan & Gehrke, 2003:4). For the past few decades, the relational database management system (RDBMS), based on the relational model introduced by Codd in 1970 (Codd, 1970; Codd, 1982), is the leading technology in the field of DBMSs. Structured Query Language (SQL) is a well-known programming language used by RDBMSs to manage and manipulate data. SQL was developed for the original RDBMSs to form the interaction interface between the user and the data (Connolly & Begg, 2005:70; Morris *et al*., 2013:83). The relational model, RDBMS and SQL are further explored in Chapter 3.

NFM use structured data efficiently, but have unstructured data which are largely underutilised.

1.3.4 Unstructured data stores

'Not only SQL' refers to a next-generation DBMS that is seen "as non-relational, distributed, open-source and horizontally scalable" (Edlich, 2009). In the information technology (IT) field, not only SQL is also referred to as NoSQL (Leavitt, 2010; Cattell, 2011). Unlike RDBMSs, that provide table structures and SQL for complex manipulation of data, NoSQL databases are not dependent on table structures and

neither do they provide SQL for the complex manipulation of data (Moniruzzaman & Hossain, 2013:1). Improved system performance is gained by this reduction in capabilities; however, distribution is still facilitated at the same time (Cattell, 2011).

NoSQL data stores support various data models that are unique and different from RDBMSs (Leavitt, 2010; Cattell, 2011; Moniruzzaman & Hossain, 2013). The main difference between the two DBMSs technologies is that NoSQL mostly stores unstructured data which are more complicated to interpret. The data stores of RDBMSs are relational and store structured data. Typical data models used in NoSQL DBMSs include key-value stores, document stores, column-oriented stores and graph databases (Cattell, 2011:12; Hecht & Jablonski, 2011:337; Moniruzzaman & Hossain, 2013:4). The data models and distribution properties of unstructured data stores are explored in Chapter 4.

NFM currently does not have the capacity to store or utilise their unstructured data. The first step in developing such a capability is to implement a NoSQL data store.

1.4 RESEARCH OBJECTIVES

The aim of this study is to design and develop an artefact that can analyse NoSQL document data stores in order to present acceptable and usable output for a user. The selected user for this study is NFM.

The research question of this study is:

Is it possible to design and develop an artefact that could analyse NoSQL document data stores in terms of relational data model constructs?

1.4.1 Objectives of the study

The following objectives have been formulated to support the research question for this study:

1.4.1.1 Primary objective

The primary objective of this study is to develop an artefact which analyses document data stores in terms of structured data model constructs.

1.4.1.2 Theoretical objectives

In order to achieve the primary objective, the following theoretical objectives were formulated for the study:

1. Gain an understanding of design science research.
2. Gain an understanding of structured data stores by focusing on the relational data model, its constructs and its characteristics.
3. Gain an understanding of unstructured data stores, such as NoSQL, by focusing on document data stores, their constructs and their characteristics.

1.4.1.3 Empirical objectives

In accordance with the primary objective of this study, the following empirical objectives were formulated:

1. Develop an artefact that could assist in analysing unstructured document data stores in terms of structured data model constructs.
2. Demonstrate the acceptability of the artefact using simulated data and tests.
3. Evaluate the utility of the artefact in the real application environment.
4. Determine the contributions that the development of the artefact could present which could advance the development of further artefacts and technologies that fall under the DSR paradigm.

1.5 RESEARCH METHODOLOGY

Developing an artefact falls under the design science research (DSR) paradigm (Vaishnavi & Kuechler, 2004) and is exploratory research (Gregor & Hevner, 2013:345). This study consists of a literature review and the development of an artefact according to the DSR paradigm.

The approaches of Peffers *et al.* (2008) and Vaishnavi and Kuechler (2004) are integrated to formulate a structure for the study. This study is divided into six activities, supporting the two DSR processes of design and evaluation (Hevner *et al.*, 2004; Hevner & Chatterjee, 2010:6). The Vaishnavi and Kuechler (2004) approach makes use of inner cycles for development and forms the structural support for the design, as well as the development activity applied in the Peffers *et al.* (2008) approach. Figure 1.2 illustrates the research process and the activities that will guide the research. The activities presented in Figure 1.2 are discussed in Chapter 2 in terms of their application to this study.

1.6 CHAPTER CLASSIFICATION

This study comprises of the following chapters:

Chapter 1 – Introduction: This chapter describes the rationale and motivation for this study its scope, and summarises the research methodology.

Chapter 2 – Research Methodology: This chapter reviews the design science research paradigm and its use in this study. Part of this chapter formulates the structure for this study.

Chapter 3 – Literature Review of Structured Data Stores: This chapter reviews the background and characteristics of the various concepts of structured data stores (traditional DBMSs) that are relevant to this study.

Chapter 4 – Literature Review of Unstructured Data Stores: This chapter reviews the background and characteristics of the various concepts of unstructured data stores (NoSQL DBMSs) that are relevant to this study.

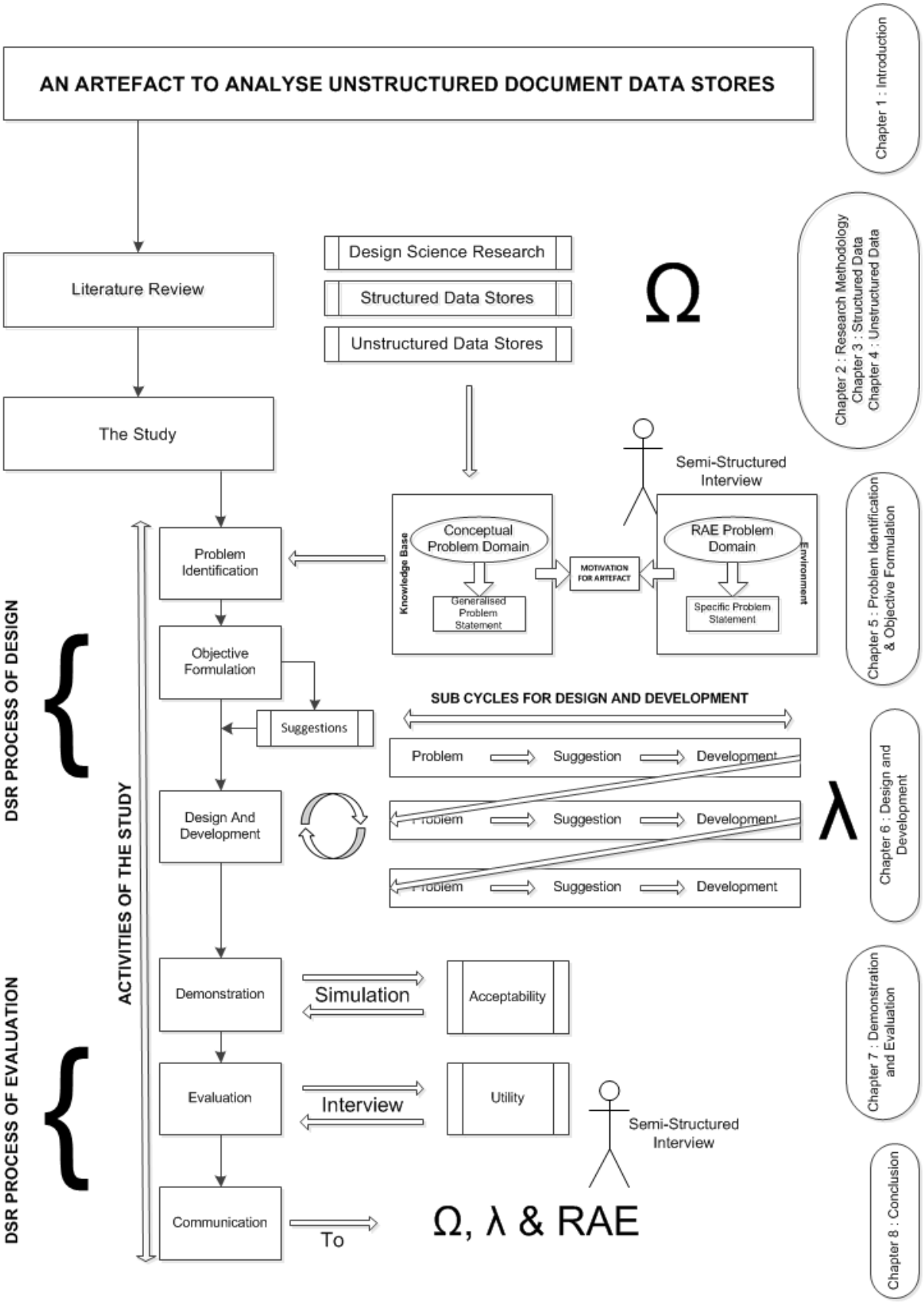


Figure 1.2: Illustration of the process followed in this study

Chapter 5 – Problem Identification and Objective Formulation: This chapter addresses the first two activities of the research methodology presented in Chapter 2. These activities include problem identification and objective formulation.

Chapter 6 – Design and Development: This chapter demonstrates the actual development of the artefact which addresses the research question and its objectives. It presents a clear structure of the development of the artefact.

Chapter 7 – Demonstration and Evaluation: This chapter addresses the fourth and fifth activities of the research methodology. These activities include demonstration and evaluation.

Chapter 8 – Communication and Conclusion: This chapter concludes with the final activity of the research methodology. It summarises the knowledge gained from this study and presents opportunities for further research.

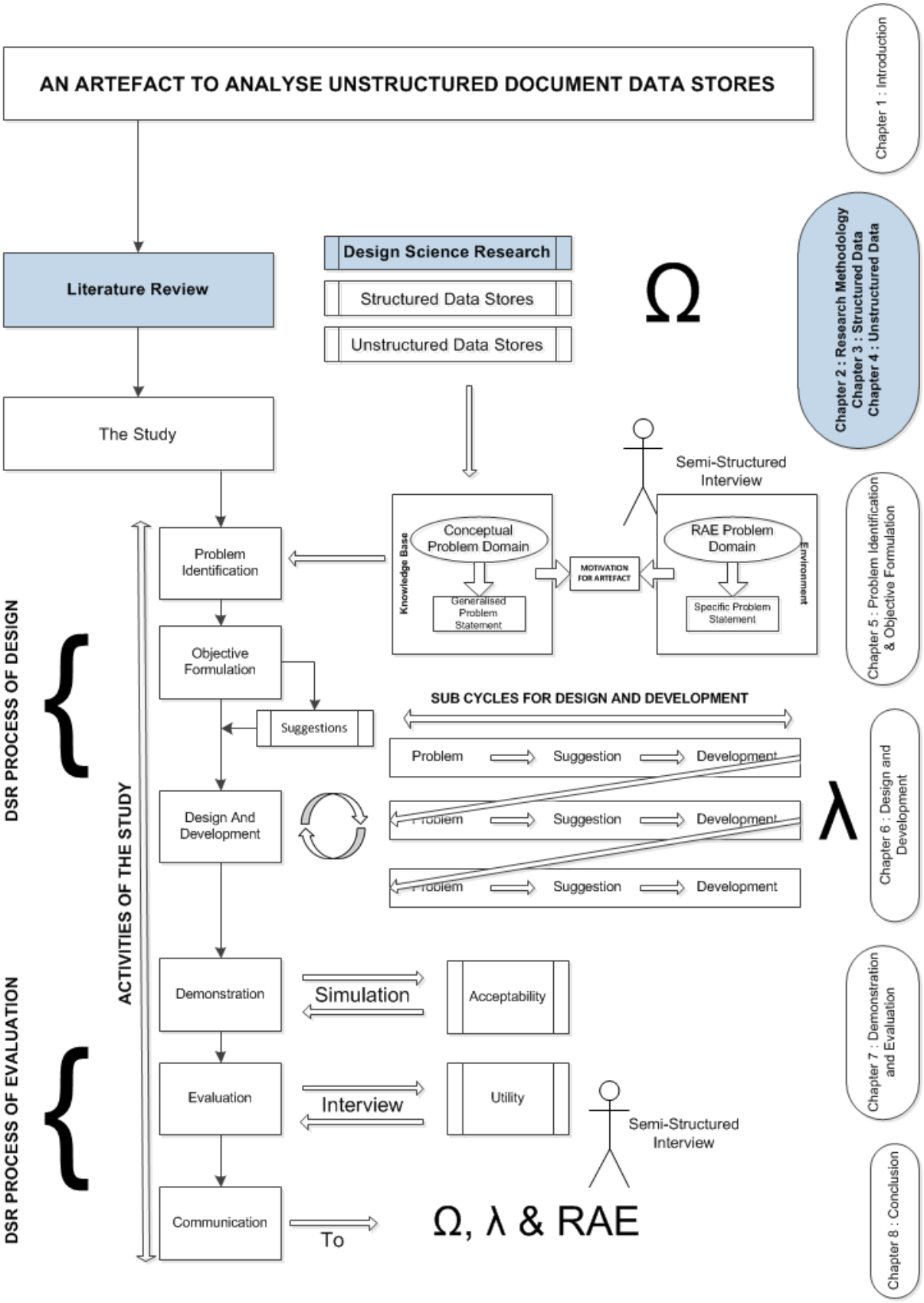
1.7 CHAPTER CONCLUSION

The objective of this chapter was to orientate this study. This objective was met by introducing the main motivation for this study, describing the aspects that are central to this study, listing research objectives, and finally presenting the chapter classification.

Addressing the problem that organisations are confronted with when utilising unstructured data is the main motivation for this study. The problem is addressed by developing an artefact which analyses unstructured data stores, specifically document data stores, and which will identify information that may be utilised by these organisations.

To assist the research process of this study, a research question, a primary objective, as well as theoretical objectives and empirical objectives, have been formulated.

The next chapter discusses the existing literature based on research methodology, with a specific focus on design science research, in addition to the formulation of the research approach and process of this study.


AN ARTEFACT TO ANALYSE UNSTRUCTURED DOCUMENT DATA STORES
Literature Review
The Study
Design Science Research
Structured Data Stores
Unstructured Data Stores
Ω
Semi-Structured Interview
Problem Identification
Objective Formulation
Suggestions
Design And Development
Demonstration
Evaluation
Communication
Knowledge Base
Conceptual Problem Domain
Generalised Problem Statement
MOTIVATION FOR ARTEFACT
RAE Problem Domain
Specific Problem Statement
Environment
SUB CYCLES FOR DESIGN AND DEVELOPMENT
Problem
Suggestion
Development
λ
Simulation
Acceptability
Interview
Utility
Semi-Structured Interview
To
Ω, λ & RAE
DSR PROCESS OF DESIGN
DSR PROCESS OF EVALUATION
ACTIVITIES OF THE STUDY
Chapter 1 : Introduction
Chapter 2 : Research Methodology
Chapter 3 : Structured Data
Chapter 4 : Unstructured Data
Chapter 5 : Problem Identification & Objective Formulation
Chapter 6 : Design and Development
Chapter 7 : Demonstration and Evaluation
Chapter 8 : Conclusion

CHAPTER TWO: RESEARCH METHODOLOGY

2.1 INTRODUCTION

The primary objective of this study is to develop an artefact which analyses document data stores in terms of structured data model constructs. In order to achieve this, a discussion of the existing literature on research methodology and design science research (DSR) is required.

In research, the contribution of knowledge is the foremost criterion for the publication of the research (Straub *et al.*, 1994:23). Research has a clear purpose – to find out things by using a systematic collection and interpretation of information (Saunders *et al.*, 2009:5). Research is conducted to acquire new knowledge by expanding frontiers in virtually all areas of science. Marczyk *et al.* (2005:1) state that, “research is often viewed as the cornerstone of scientific progress”. According to Saunders *et al.* (2009:3), the term research methodology refers to “the theory of how research should be undertaken”.

Design science research (DSR) is one of many approaches available in order to acquire new knowledge. DSR is mostly concerned with the development and application of an artefact according to a scientifically responsible methodology. The artefact is designed to acquire knowledge and understanding of an identified design problem (Hevner & Chatterjee, 2010:6). The term artefact refers to an artificial man-made object. According to Simon (1996:3), the term artificial has "a pejorative air about it that we must dispel before we can proceed". Simon (1996:4) therefore defines the artificial as "Produced by art rather than by nature; not genuine or natural; affected; not pertaining to the essence of the matter"[1]. The artificial does not occur naturally but is man-made, and the artefact, the artificial man-made object, must "serve its intended purpose" (Simon, 1996:6).

[1] Capitalisation and punctuation in direct quotations from original text were left unchanged. Grammatical changes in direct quotations are indicated by [].

The objective of this chapter is to demonstrate an understanding of research methodology and to position this study in the research framework. Research philosophies, paradigms and methods in general are discussed in this chapter. An exposition of the DSR paradigm and formulation of the research approach and process of this study follows.

This chapter is divided into the following sections: research philosophy (2.2); research paradigms (2.3); positioning the study (2.4); design science research (2.5); data collection techniques (2.6); research process of this study (2.7); and finally, the conclusion (2.8).

2.2 RESEARCH PHILOSOPHY

Saunders *et al.* (2009:128) state that research philosophy "relates to the development of knowledge and the nature of that knowledge". The research philosophy adopted in this study is influenced by practical considerations and assumptions made by the researcher. It reflects the way the researcher views the world (Saunders *et al.*, 2009:106).

Three philosophical assumptions are identified by Blanche *et al.* (2006:6), namely: ontological, epistemological and methodological assumptions. Axiological is a fourth type of philosophical assumption identified by Vaishnavi and Kuechler (2004). Ontological assumptions are concerned with the nature of reality, while epistemological assumptions are concerned with what constitutes reasonable knowledge in a field of study. Methodological assumptions concern the process the researcher uses to gain knowledge within the field of study. Axiological assumptions refer to things the researcher believes to be of value in relation to the study. Axiology is important in the DSR paradigm, since the researcher has control over the creation and understanding of a designed artefact, and therefore the impact that the artefact has in a complex problem environment.

A discussion of research paradigms follows in order to position this study.

2.3 RESEARCH PARADIGMS

There are three classical research paradigm types identified by Blanche *et al.* (2006:6), namely: positivist, interpretivist and constructionist research. Some of these paradigms are labelled differently by information systems (IS) research groups. The constructionist paradigm is labelled at times as critical social research (Adebesin *et al.*, 2011:311).

Positivists posit that knowledge is gained by experience of reality through the senses (Noor, 2008:1602). In essence, the researcher will be working with an objectively observable reality. The reality is observed and data are collected using the senses. Data analysis is done according to statistical analysis methods, focusing on relationships between variables. The result may be law-like generalisations (Remenyi *et al.*, 1998:32). Quantitative (numerical) data are most often used in positivist studies (Saunders *et al.*, 2009:119). Experiments and hypotheses testing are the general methodology employed in quantitative research.

Interpretivists posit that it is important to understand the differences in humans' roles as social actors (Saunders *et al.*, 2009:116). This paradigm is mostly employed by social science researchers, as opposed to positivism, which is mostly applied by the natural scientist. In essence, interpretivism goes beyond facts towards meaning (Noor, 2008). The emphasis is placed on conducting research among humans, instead of conducting research on objects such as trucks and computers (Saunders *et al.*, 2009:113). Qualitative (narrative) data are mostly used in interpretive studies.

Constructionists or *critical social researchers* posit that reality is socially constructed, and an individual's construct thereof is influenced by societal norms (Myers, 1997). Fundamentally, the researcher is not detached from the subjects of study; therefore the interpretation of an event is influenced by the researcher's personal, cultural and historical experiences. The constructionist regularly addresses the process of interaction between individuals (Creswell, 2003:9). The methodology regularly implemented by IS constructionists is known as Action Research (AR) accredited to Lewin (1946).

AR has been defined by Rapoport (1970:499) as follows:

> *"Action research aims to contribute both to the practical concerns of people in an immediate problematic situation and to the goals of social science by joint collaboration within a mutually acceptable ethical framework".*

Design science research (DSR) is a fourth research paradigm which changes the state of the world through the introduction of novel artefacts (Vaishnavi & Kuechler, 2004). The DSR paradigm is defined by Hevner *et al.* (2004:76) in information systems as:

> *"A problem-solving paradigm which seeks to create innovations that define ideas, practices, technical capabilities, and products through which the analysis, design, implementation, management, and use of information systems can be effectively and efficiently accomplished."*

DSR has had an increasing acceptance as a legitimate approach to Information Systems (IS) research (Gregor & Hevner, 2013:337).

Baskerville (2008:442) clearly states that "design science is not action research" and that "action research is clearly centred on discovery through action", while "design science is clearly centred on discovery through design". DSR focuses on the designed artefact and not the human interaction towards the artefact as in the case with AR. Iivari and Venable (2009:4) state: "When compared with AR, an essential difference is that DSR assumes neither any specific client nor joint collaboration between researcher and the client".

Table 2.1 summarises the main features and philosophical assumptions of positivist, interpretive, constructionist and design research paradigms.

Table 2.1: Research paradigms and their philosophical assumptions adapted from Adebesin et al. (2011:310), Blanche et al. (2006) and Vaishnavi and Kuechler (2004)

	PHILOSOPHICAL ASSUMPTIONS			
RESEARCH PARADIGMS	***Ontology***	***Epistemology***	***Methodology***	***Axiology***
Positivist	• Single, stable reality • Law-like	• Objective • Detached observer	• Experimental • Quantitative • Hypothesis testing	• Truth • Prediction
Interpretive	• Multiple realities • Socially constructed	• Empathetic • Observer subjectivity	• Interactional • Interpretation • Qualitative	• Contextual understanding
Constructionist/Critical social theory	• Socially constructed reality • Discourse • Power	• Suspicious • Political • Observer constructing • Versions	• Deconstruction • Textual analysis • Discourse analysis	• Inquiry is value-bound • Contextual understanding • Researcher's values affect the study
Design Science Research	• Multiple, contextually situated realities	• Knowing through making • Context-based construction	• Developmental • Impact analysis of artefact on composite system	• Control • Creation • Understanding

2.4 POSITIONING THE STUDY

Positioning the study within one of four paradigms helps the researcher to select appropriate methods to use for the study. In this study it is believed that the value of the artefact is important to the study, and therefore the philosophical position taken is that of Design Science Research. The control, creation and understanding of the designed artefact are important. What supports this placement is that a key distinguishing feature of DSR is the utilisation of a specific artefact to address a specific business problem (Gregor & Hevner, 2013:342). Therefore applying the artefact at Newcom Fluid Management (NFM) makes this paradigm most appropriate.

Since this study makes extensive use of DSR, it is important to discuss this paradigm comprehensively.

2.5 DESIGN SCIENCE RESEARCH

Design science research (DSR) has been put into practise for some time in the engineering and Information Systems (IS) disciplines (Gregor & Hevner, 2013:338). DSR is defined within the IS discipline as the construction of a wide range of socio-technical artefacts such as decision support systems, modelling tools, governance strategies, methods of IS evaluations and IS change interventions (Gregor & Hevner, 2013:337). DSR also analyses the performance of a designed artefact in order to understand and improve the artefact (Vaishnavi & Kuechler, 2004; Hevner & Chatterjee, 2010:30). DSR is primarily the creation and evaluation of an artefact used to acquire the solution to the identified organisational problem through understanding thereof (Hevner *et al.*, 2004:82; Hevner & Chatterjee, 2010:6). The evaluation of these artefacts could be subject to quantitative and/or empirical and qualitative methods (Hevner *et al.*, 2004:77).

The sections that follow give an overview of DSR. These sections include concepts central to DSR; the process of DSR; DSR approaches; and the guidelines available for practicing and evaluating DSR.

2.5.1 Concepts central to design science research

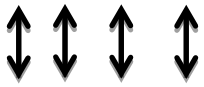
The concepts central to DSR are: the role of knowledge and the knowledge contributions framework.

2.5.1.1 Design science research knowledge

The knowledge contribution made by DSR is effective when it is clear and related to the real-world application environment (RAE) from where the research problem or opportunity is drawn (Hevner *et al.*, 2004). The RAE refers to industry and academic fields, but it is not limited to these fields. Research conducted in DSR contributes to

knowledge and generalised theory (Hevner *et al.*, 2004; Vaishnavi & Kuechler, 2004; Gregor, 2006; Gregor & Jones, 2007). DSR contributions can be made on three maturity levels that were built on a framework introduced by Purao (2002). Table 2.2 illustrates DSR artefact types and provides an example of each level of maturity.

Table 2.2: DSR Contribution Types (Gregor & Hevner, 2013:342)

	CONTRIBUTION TYPES	EXAMPLE ARTEFACTS
More abstract, complete, and mature knowledge	Level 3. Well-developed design theory about embedded phenomena	Design theories (mid-range and grand theories)
	Level 2. Nascent design theory—knowledge as operational principles/architecture	Constructs, methods, models, design principles, technological rules.
More specific, limited, and less mature knowledge	Level 1. Situated implementation of artefact	Instantiations (software products or implemented processes)

The knowledge of a DSR project should include reference to a kernel theory (Gregor & Hevner, 2013:340). According to Walls *et al.* (1992:48), kernel theory refers to "theories from natural science, social sciences and mathematics". The reason for the inclusion of kernel theories in DSR knowledge is to explain why the design works (Gregor & Hevner, 2013:340). For the purpose of this study, kernel theory guides the artefact's creation and refers to prescriptive knowledge.

DSR knowledge can be divided into two distinct types known as descriptive knowledge (denoted Ω or omega) and prescriptive knowledge (denoted λ or lambda). Descriptive knowledge is known as the 'what' knowledge about natural phenomena and the laws as regularities among phenomena, while prescriptive knowledge is the 'how' knowledge of a human-built artefact (Gregor & Hevner, 2013:343). Figure 2.1 shows the knowledge base for a DSR domain.

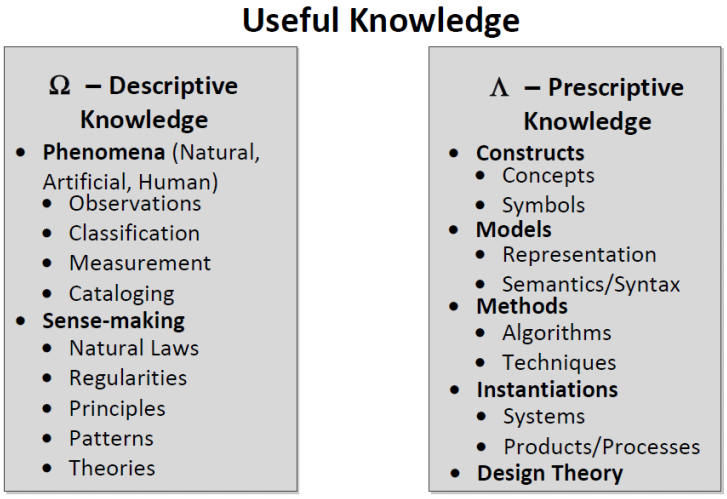


Figure 2.1: DSR knowledge base (Gregor & Hevner, 2013:344)

Hevner and Chatterjee (2010) and Iivari (2007) state that DSR begins in the application environment when an opportunity, challenging problem or insightful vision for something innovative is presented. The success of a DSR project rests on the research skills of the research team, who aim to draw knowledge appropriately from both descriptive and prescriptive sources. The relationships and interactions of descriptive and prescriptive knowledge are key insights into the performance of DSR. Figure 2.2 illustrates these relationships and interactions and the roles DSR plays in the application environment with reference to descriptive and prescriptive knowledge.

A DSR project has the potential to make many different types and levels of research contributions. These contributions depend on the starting points in terms of problem maturity and solution maturity. It should be noted that there is limited advice on how to signal and assess the degree of the contribution made by a DSR project in IS literature to date. The nature of the artefact often presents difficulty in identifying a knowledge contribution (Gregor & Hevner, 2013:340). The DSR knowledge contribution framework (KCF), discussed in the following section, helps us to understand and position the contribution of a DSR project.

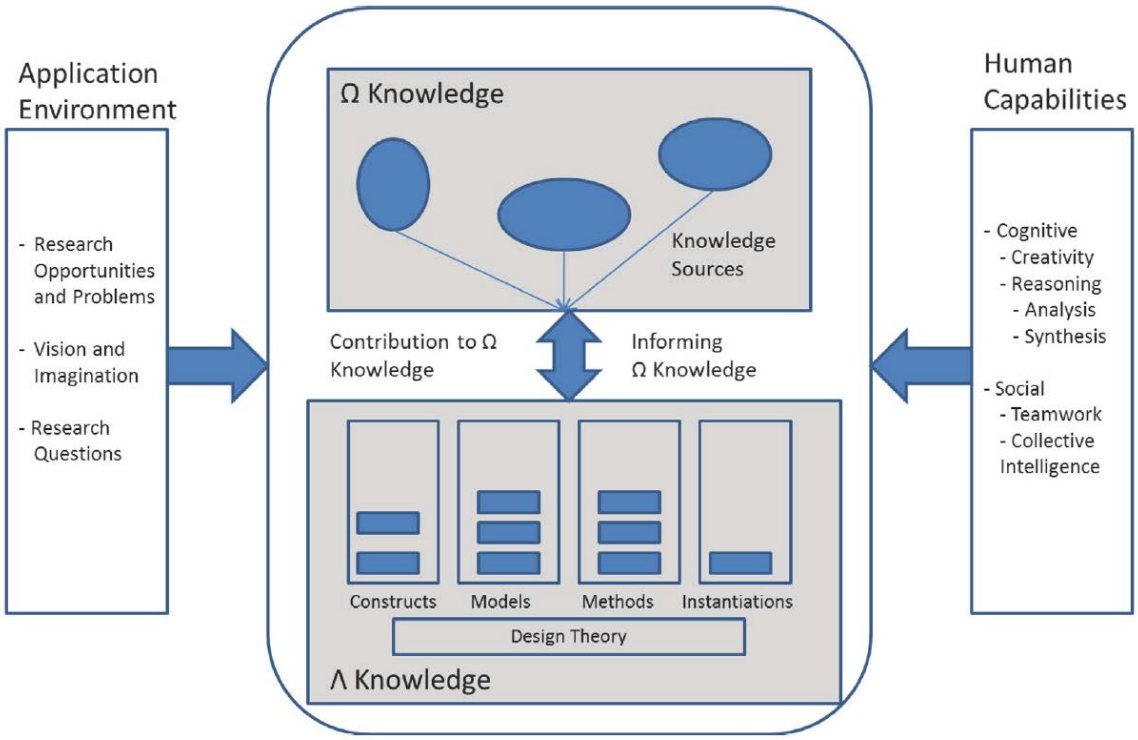


Figure 2.2: DSR knowledge roles (Gregor & Hevner, 2013:344)

2.5.1.2 The Knowledge Contribution Framework

The degree of knowledge can vary from incremental artefact construction to partial theory building, but may still be a significant and publishable contribution (Gregor & Hevner, 2013:343). The DSR KCF is divided into four quadrants: improvement, invention, routine design and exaptation. Figure 2.3 displays a summative description and location of knowledge within the framework. The four quadrants are briefly discussed.

The *improvement* quadrant deals with the development of a new solution to a known problem. The goal is the creation of an efficient and effective solution for products, processes, services, technologies or ideas. The presentation of solutions from this quadrant should clearly indicate how the new solutions differ from the current solutions (Gregor & Hevner, 2013:346).

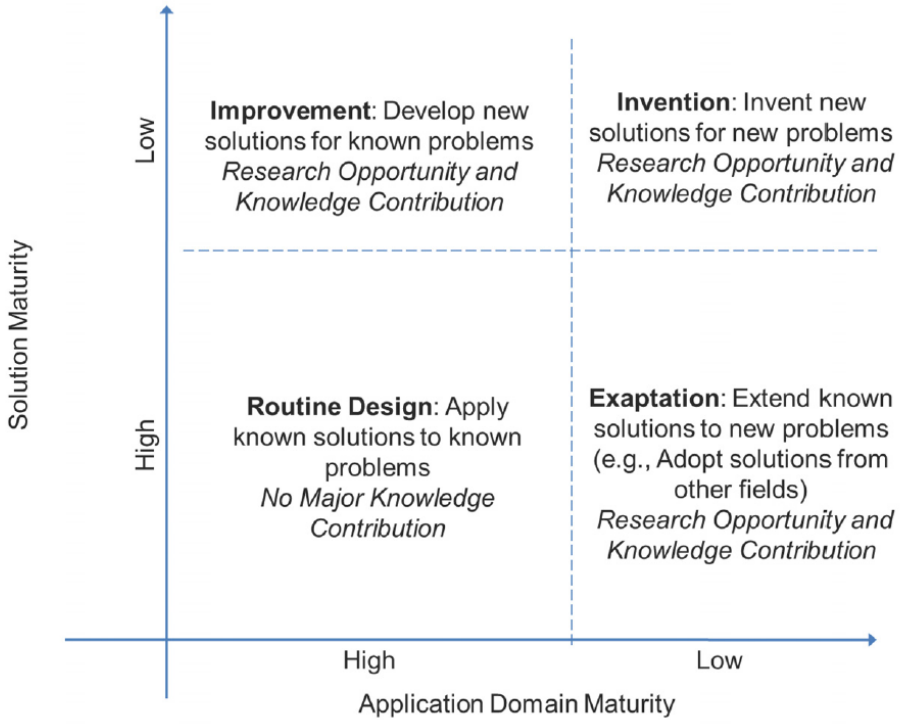


Figure 2.3: DSR Knowledge Contribution Framework (Gregor & Hevner, 2013:345)

The *invention* quadrant deals with the construction of a new solution to an arising problem. The process of invention is described as an exploratory search in the context of a complex problem space requiring the cognitive skills of curiosity, imagination, creativity, insight and knowledge of multiple realms of enquiry to find a feasible solution. The activities found within invention can be considered as DSR when the result is an artefact that can be applied and evaluated in a real-world context (Gregor & Hevner, 2013:345). A DSR project found within this quadrant will entail conducting research revolving around new and interesting applications. Within these application environments, little understanding of the problem context exists, and no prior solutions are available. The fact is that so little is known about the problem that the research question has not been raised. The interestingness of the research is what guides the study (Simon, 1996:162), and the recognised problem may not necessarily exist and therefore solutions may be unclear (Gregor & Hevner, 2013:346).

The *routine* design quadrant deals with the application of an existing solution to a known set of problems. The existing knowledge about the problem is well understood and the artefact is used to address the problem (Gregor & Hevner, 2013:346). Work in this quadrant is normally not thought of as a contribution, but surprises and discoveries may occur in some cases (Stokes, 1997). In cases like these, the research will likely move towards other quadrants (Gregor & Hevner, 2013:347).

Finally, the *exaptation* quadrant extends known solutions to new problems. Exaptation is common in IS, where new technologies lead to new applications. The presentation of exaptation is that the researchers need to demonstrate that the extension of existing knowledge towards a new field is of some importance and interest (Gregor & Hevner, 2013:347). By applying existing knowledge of structured data into the new field of unstructured data, this study may be positioned within the exaptation quadrant.

The KCF should be seen as a guide for researchers to follow, but it should not be followed blindly. It is not appropriate or useful for researchers to force results into a quadrant or design theory description (Gregor & Hevner, 2013:352).

The following section provides DSR methodologies within IS research and assists in formulating the process for the study.

2.5.2 Design science research process

The DSR process consists of two main processes in the IS research cycle. The two processes are concerned with the design of the IT artefact intended to solve the problem and the evaluation thereof. Hevner *et al.* (2004:78) state that “design is both a process and a product”, where the process is the set of activities used to create the product which is the actual artefact. The evaluation provides feedback about the artefact to improve the artefact and gain a better understanding of the problem context. According to Hevner *et al.* (2004:78), further evaluation “enable[s]

researchers to learn about the real world, how the artefact affects it, and how users appropriate it". Design and evaluation form a cycle that is iterated a number of times until the artefact is finalised (Markus *et al.*, 2002).

The cycle of design and evaluation is illustrated in the centre of Figure 2.4. Figure 2.4 illustrates that knowledge is drawn from both the knowledge base and the RAE. The problem drawn from the RAE (Iivari, 2007; Hevner & Chatterjee, 2010), referring to industry or academia, is illustrated in the left block of Figure 2.4. The prescriptive and descriptive knowledge is illustrated in the right block of Figure 2.4. The outcome and evaluation of the project in turn feed back towards the knowledge base and the RAE.

2.5.3 Design science research approaches

There are several approaches available to guide researchers in performing DSR research. The approaches suggested by Peffers *et al.* (2008) and by Vaishnavi and Kuechler (2004) are key to this study and the combination of these approaches is presented in Section 2.7.

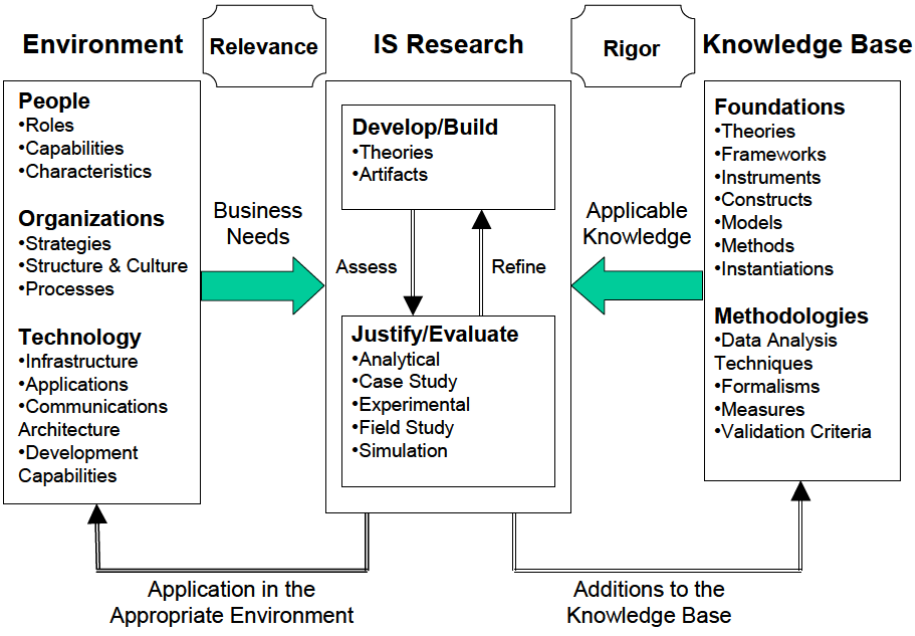


Figure 2.4: IS Research Framework (Hevner et al., 2004:80)

The approach to DSR suggested by Peffers *et al.* (2008) contains six activities: the first activity is the identification and motivation of a relevant research problem; the second activity involves the definition of the solution objectives; the third activity is the design and development of DSR artefacts; the fourth activity is the demonstration of the artefacts' ability to solve one or more problems; the fifth activity is the evaluation of the created artefacts and the sixth activity is the communication of the research. Peffers *et al.* (2008:56) state that "This process is structured in a nominally sequential order; however, there is no expectation that researchers would always proceed in sequential order from activity 1 through activity 6. In reality, they may actually start at almost any step and move outward". Figure 2.5 demonstrates the Peffers *et al.* (2008:53) process model and Table 2.3 describes these activities.

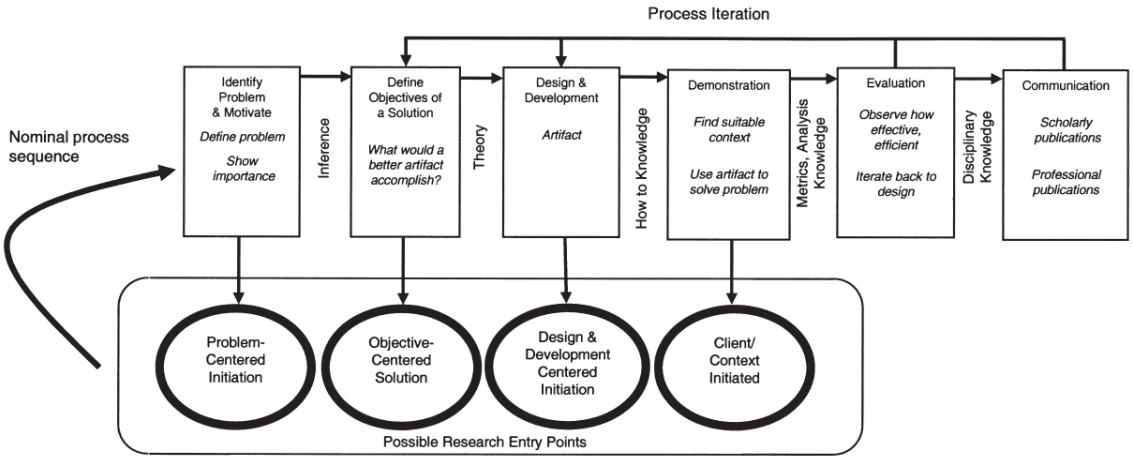


Figure 2.5: DSR Process Model (Peffers et al., 2008:53)

Table 2.3: DSR activities summarised from Peffers et al. (2008:52-56)

ACTIVITY	DESCRIPTION

Activity 1: Problem identification and motivation	• Define the specific research problem.
	• Justify the value of a solution, because it:
	  o Motivates the researcher and the audience of the research to pursue the solution and to accept the results.
	  o Helps to understand the reasoning associated with the researcher's understanding of the problem.
	• Resources required: knowledge of the state of the problem; and importance of its solution.

Activity 2: Define the objectives for a solution	• Deduce the objectives of a solution from the problem definition. • Deduce the knowledge of what is possible and feasible. • The objectives can be: o quantitative or qualitative • The objectives should be inferred rationally from the problem specification. • Resources required: knowledge of the state of problems; knowledge of current solutions, if any, and their efficacy.
Activity 3: Design and development	• Create the artefact. Conceptually, a design research artefact can be any designed object in which a research contribution is embedded in the design. • Determine the artefact's desired functionality. • Determine the artefact's architecture. • Resources required: knowledge of theory that can be used to address the problem situation.

Activity 4: Demonstration	• Demonstrate the use of the artefact to solve one or more instances of the problem.
	• Could involve its use in:
	  o experimentation,
	  o simulation,
	  o case study,
	  o proof, or
	  o other appropriate activities.
	• Resources required: effective knowledge of how to use the artefact to solve the problem.

Activity 5: Evaluation	• Observe and measure how well the artefact supports a solution to the problem. • Compare the objectives of a solution to actual observed results

	from use of the artefact in the demonstration. • Iterative in nature. • Resources required: knowledge of relevant metrics and techniques.
Activity 6: Communication	• Communicate: o the problem and its importance, o the artefact, its utility and novelty, o the rigor of its design, and o effectiveness to researchers and other relevant audiences. • Resources required: knowledge of the disciplinary culture.

The approach by Vaishnavi and Kuechler (2004) to DSR consists of five phases: awareness of the problem, suggestions, development, evaluation and conclusion. The five phases suggested by Vaishnavi and Kuechler (2004) form the main outer cycle of a DSR study. The development phase may be subdivided into inner cycles of repetitive phases. The first and main outer cycle presents the overall objective of the development of the artefact and the second and inner cycle presents the detailed steps of creating the artefact. These phases are depicted in Figure 2.6 and described in Table 2.4.

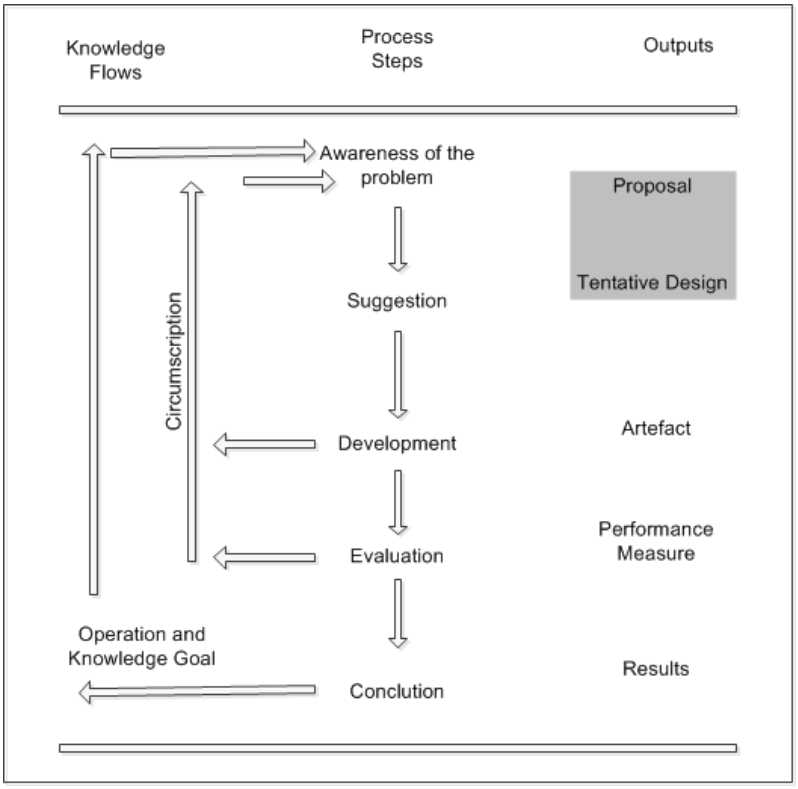


Figure 2.6: Design research phases by Vaishnavi and Kuechler (2004)

Table 2.4: DSR phases summarised from Vaishnavi and Kuechler (2004)

PHASE	DESCRIPTION
Awareness of Problem	• The awareness may be drawn from multiple sources such as: o new developments in industry, o a reference discipline, and o reading in an allied discipline. • Output: proposal, formal or informal, for a new research effort.
Suggestion	• This phase follows a proposal and is connected to it. • Output: tentative design.
Development	• The tentative design is implemented. • Techniques for implementation vary depending on the artefact to be constructed. • Output: implemented artefact.
Evaluation	• The artefact is evaluated according to criteria. • Output: deviations from expectations, both quantitative and

	qualitative noted and tentatively explained.
Conclusion	• Final phase of a specific research effort. • Is the result satisfactory? That is: o though there are still deviations in the behaviour of the artefact from the (multiple) revised hypothetical predictions, the result is acceptable. • Output: artefact that is 'good enough'.

2.5.4 Design science research guidelines

DSR is a problem-solving process. The fundamental principle of DSR is that knowledge is acquired by understanding the design problem and that the solution is acquired in the building and application of the artefact (Hevner *et al.*, 2004:82). The guidelines for practicing DSR presented here are subject to the researcher's creative skills and judgement to determine when, where and how to apply each guideline in a research project (Hevner *et al.*, 2004:82). In order for a DSR project to be complete, it is essential that each guideline is addressed in some manner (Hevner *et al.*, 2004:82). Table 2.5 is a summary of these guidelines presented by Hevner *et al.* (2004:82).

Table 2.5: DSR research guidelines quoted from Hevner et al. (2004:83)[2]

GUIDELINE	DESCRIPTION
Guideline 1: Design as an Artefact	Design-science research must produce a viable artefact in the form of a construct, a model, a method or an instantiation.
Guideline 2: Problem Relevance	The objective of design science research is to develop technology-based solutions to important and relevant business problems.
Guideline 3: Design Evaluation	The utility, quality and efficacy of a design

[2] "Design-science research" is directly quoted from the source and is therefore written with a hyphen, in contrast to the rest of the study.

	artefact must be rigorously demonstrated via well-executed evaluation methods.
Guideline 4: Research Contributions	Effective design-science research must provide clear and verifiable contributions in the areas of the design artefact, design foundations, and/or design methodologies.
Guideline 5: Research Rigor	Design-science research relies upon the application of rigorous methods in both the construction and evaluation of the design artefact.
Guideline 6: Design as a Search Process	The search for an effective artefact requires utilizing available means to reach desired ends while satisfying laws in the problem environment.
Guideline 7: Communication of Research	Design-science research must be presented effectively both to technology-oriented as well as management-oriented audiences.

These guidelines formulate a benchmark against which a DSR project can be effectively evaluated. It also provides a foundation for the researcher to do self-reflection at the end of the DSR project. The questions provided in Table 2.6 by Hevner and Chatterjee (2010:20) constitute a checklist for a DSR project:

Table 2.6: DSR checklist quoted from Hevner and Chatterjee (2010:20)

#	QUESTIONS
1	What is the research question (design requirements)?
2	What is the artefact? How is the artefact represented?
3	What design processes (search heuristics) will be used to build the artefact?
4	How are the artefact and the design processes grounded by the knowledge base? What, if any, theories support the artefact's design and the design process?

5	What evaluations are performed during the internal design cycles? What design improvements are identified during each design cycle?
6	How is the artefact introduced into the application environment and how is it field-tested?
7	What metrics are used to demonstrate artefact's utility and improvement over previous artefacts?
8	What new knowledge is added to the knowledge base and in what form (e.g. peer-reviewed literature, meta-artefacts, new theory and new method)?
9	Has the research question been satisfactorily addressed?

This checklist is used in Chapter 8 to reflect on the study.

The next section discusses data collection techniques applied in this study, which may be used to guide problem identification and evaluation.

2.6 DATA COLLECTION TECHNIQUES

There are various techniques available for a researcher to collect data. Some of the common techniques include: interviews, questionnaires, observations and documents (Oates, 2006:116; Saunders *et al.*, 2009:146). Table 2.7 describes the data collection techniques.

Table 2.7: Data collection techniques quoted from Saunders et al. (2009:146)

TECHNIQUES	DESCRIPTIONS
Interviews	• A purposeful discussion between two or more people.
Questionnaires	• Include all techniques of data collection in which each person is asked to respond to the same set of questions in a predetermined order.
Observations	A systematic: • observation, • recording,

	• description, • analysis and • interpretation of people's behaviour.
Documents	Includes written documents similar to: • notices, • minutes of meetings, • diaries, • administrative and public records and • reports to shareholders. Further includes non-written documents similar to: • voice and video recordings, • pictures, • films and television programmes.

The data collected through the application of these techniques may be grouped into quantitative data and qualitative data.

Quantitative data are defined by Saunders *et al.* (2009:151) as:

> *"A synonym for any data collection technique (such as a questionnaire) or data analysis procedure (such as graphs or statistics) that generates or uses numerical data."*

However, qualitative data are defined by Saunders *et al.* (2009:151) as:

> *"A synonym for any data collection technique (such as an interview) or data analysis procedure (such as categorising data) that generates or uses non-numerical data."*

Seaman (1999:572) argues that “qualitative data is richer than quantitative data”. Data collected using qualitative methods contain more information than data collected using quantitative methods (Seaman, 1999:572).

Data collection techniques may be combined at times by the researcher. The combination of qualitative and quantitative data are known as mix-method techniques and referred to as triangulation (Saunders *et al.*, 2009:146).

According to Gregor and Hevner (2013), qualitative methods may be used in the evaluation of an artefact. Data for the evaluation of the artefact in this study were collected by conducting interviews. Qualitative data were gathered and analysed using interpretive content analysis (refer to Section 2.6.2).

Since interviews were used in this study, a description of interviews follows next.

2.6.1 Interviews

An interview is described by Kahn and Cannell (1957) as a purposeful discussion between two or more people. Valid and reliable data may be gathered using interviews for research purposes (Saunders *et al.*, 2009:318). There are several types of interviews available: structured interviews; semi-structured interviews; and unstructured or in-depth interviews. The types of interviews are described in Table 2.8.

Table 2.8: Interview types quoted from Saunders et al. (2009:320)

TYPE	DESCRIPTION
Structured interviews	• Questionnaires based on a predetermined and standardised set of questions. • An identical set of questions. • Interviewer-administered questionnaires.

Semi-structured interviews	• List of themes and questions to be covered. • May vary from interview to interview.
Unstructured interviews	• Informal. • Explore a general area in depth. • Area of interest.

Both structured and semi-structured interviews were used in this study. A structured interview is used for the evaluation of the artefact by the RAE. A semi-structured interview is used for problem identification and requirements gathering. To help organise both types of interview, the researcher may make use of interview guidelines.

2.6.1.1 Interview guidelines

Planning and preparation are important requirements for an interview. This enables the establishment of goals and a plan to account for practical issues. Rogers *et al.* (2011) recommend a pilot study to uncover any problems and gain essential experience. Rogers *et al.* (2011:390-391) propose several guidelines for conducting an interview. Table 2.9 summarises these guidelines.

Table 2.9: Guidelines for conducting interviews quoted from Rogers et al. (2011:390-391)

GUIDELINES	REASON
Avoid long questions.	Difficult to remember.
Compound questions should be deconstructed into separate questions.	Could be confusing.
Do not use technical terms.	Not always fully understood by the interviewee.
Leading questions should be avoided.	This presupposes a particular response from interviewees.
Biases should not be reflected in	Interviewers must be aware of own personal biases.

questions posed.

Rogers *et al.* (2011:391) suggest several stages for conducting an interview. Table 2.10 summarises these stages.

Table 2.10: Stages during an interview quoted from Rogers et al. (2011:391)

STAGE	REASON
Introductory session	The interviewer: • introduces himself/herself; • explains the purpose of the interview; • reassures the interviewee; • explains how the data collected will be used; and • seeks the interviewee‘s permission for the use of the data.
Warm-up session	• The interviewer asks simple, general questions, e.g. o Demographic questions.
Main session	• The interviewer asks questions that address the main focus of the interview. • Questions are posed in ways that depend on the response of each participant.
Cooling-off period	• The interviewer asks further simple questions. o This eases any tension. • An opportunity is given for interviewees to provide any information they wish to add.
Closing session	• The interviewer thanks the interviewee for participating.

According to Rogers *et al.* (2011:391) professionalism is the golden rule when conducting interviews. The following list provides advice for conducting an interview quoted from Rogers *et al.* (2011:391):

- “dress similar to interviewees if possible or dress neatly and avoid standing out;
- prepare an informed consent form for signing;
- if recordings are used, test it prior and make sure how to use it properly; and

- record answers exactly."

The qualitative data of this study is analysed using qualitative data analysis.

2.6.2 Qualitative data analysis

The qualitative data analysis process is defined by Bogdan and Biklen (1982:152) as:

> "…*working with data, organizing it, breaking it into manageable units, synthesizing it, searching for patterns, discovering what is important and what is to be learned, and deciding what you will tell others.*"

The placing of raw data into logical, meaningful units of information requires some creativity in the process of qualitative data analysis (Hoepfl, 1997). This process of qualitative data analysis consists of three stages: the identification stage, the re-examination state and the translation stage.

The process of qualitative data analysis begins with the identification stage – meaningful themes that emerge from the raw data (Hoepfl, 1997). Strauss and Corbin (1990) refer to this process as 'open coding'. The process of open coding involves the identification and the naming of conceptual themes by the researcher (Hoepfl, 1997). The phenomena observed will be grouped into these themes. The goal, according to Hoepfl (1997), for theme identification is to "create descriptive, multi-dimensional categories which form a preliminary framework for analysis". Hoepfl (1997) recommends that researchers devise an 'audit trail' by identifying data chunks to their speakers and the context.

The next stage of qualitative data analysis is the re-examination stage. During this stage, connections between the identified themes are determined by means of re-examination (Hoepfl, 1997). This is a complex process developed by Strauss and

Corbin (1990) and is referred to as ‘axial coding’. The combination and comparison of the themes identified form the big picture (Hoepfl, 1997).

The final stage, or translation stage, of qualitative data analysis involves the conversion of the conceptual model into a story line. Strauss and Corbin (1990:57) state that it should “closely approximate the reality it represents”. Hoepfl (1997) explains that regardless of the fact that these stages are linear, they may occur simultaneously or repeatedly. Hoepfl (1997) also states that further data collection may occur during any stage should gaps be found within the data.

The interviews in this study were subjected to qualitative data analysis. The objective of the qualitative data analysis of the first interview was to determine the problem of the RAE. The objective of the qualitative data analysis of the second interview was to evaluate the artefact by the RAE. Coding was used in both instances to tabulate and group the results.

The next section discusses the research methodology of this study and presents the structure.

2.7 RESEARCH PROCESS OF THE STUDY

The approaches of Peffers *et al.* (2008) and Vaishnavi and Kuechler (2004) were integrated to formulate a structure for the study.

The Vaishnavi and Kuechler (2004) approach makes use of inner cycles for development and forms the structural support for the design as well as the development activity applied in the Peffers *et al.* (2008) approach. Figure 2.8 illustrates the study and the activities that guided the researcher through the study.

As discussed in Section 2.5.1.1, knowledge used in DSR projects can be categorised as either prescriptive or descriptive knowledge. Figure 2.7 is an adaptation of Figure 2.2 indicating the layout of the study and illustrating where the

literature fits within the DSR knowledge roles. The RAE is illustrated in the left block of Figure 2.7. The RAE here refers to NFM from where the problem was drawn. The prescriptive knowledge (denoted λ or lambda) forms the centre lower block of the figure and is used to guide the development of the artefact as discussed in Chapters 5 and 6. The descriptive knowledge (denoted Ω or omega) forms the centre upper block of the figure. This knowledge is drawn from the literature based on structured and unstructured data stores discussed in Chapters 3 and 4. The human capabilities are illustrated in the right block of the figure and are drawn from the researcher's prior knowledge and experience.

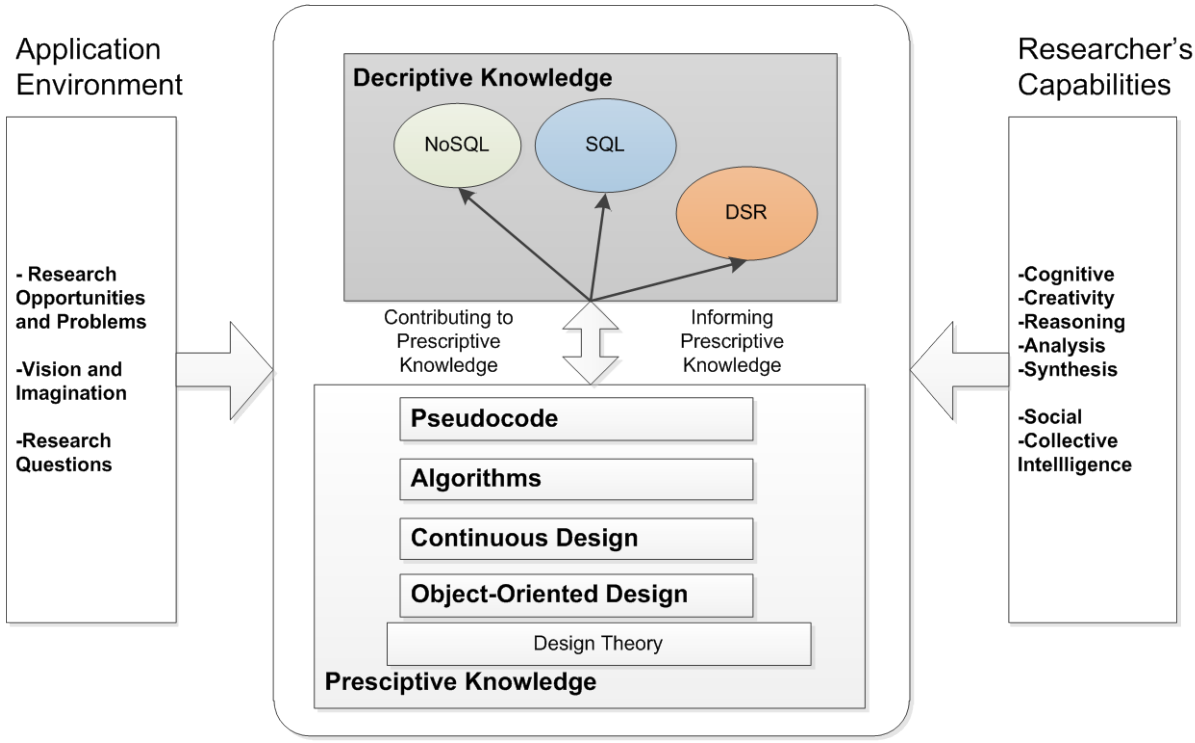


Figure 2.7: DSR knowledge and relations and interactions for the study

The activities presented in Figure 2.8 are discussed in terms of their application in this study.

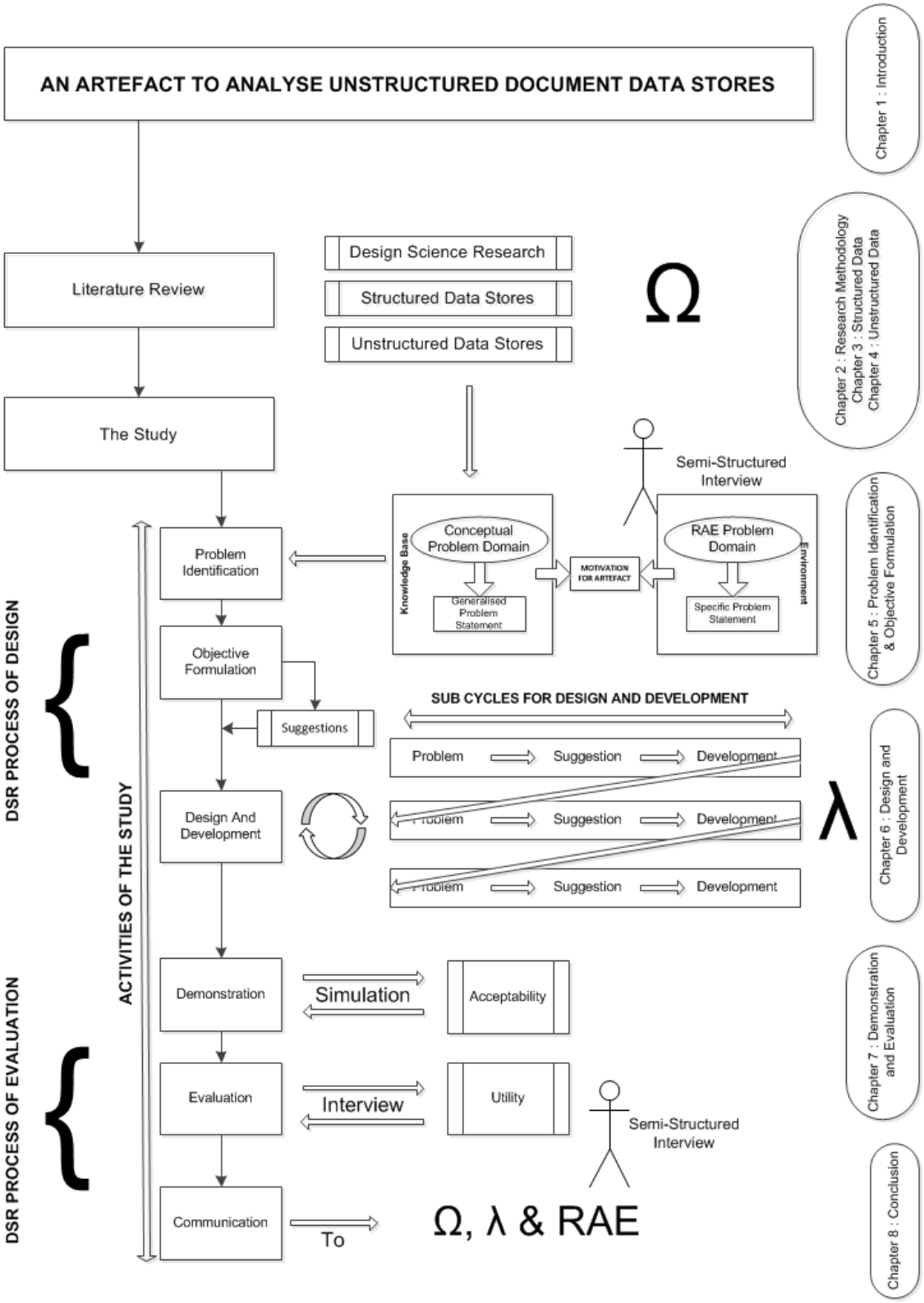


Figure 2.8: Illustration of the process followed in this study.

2.7.1 Problem identification

Problem identification activity of this DSR study puts the problem domain in context. The problem is drawn from two domains, namely a conceptual idea and a real application environment (RAE).

The first domain is drawn from the conceptual idea of the problem and is supported by the literature from Chapters 3 and 4.

The second domain is drawn from the RAE. The RAE representative is interviewed and qualitative data are gathered. Qualitative data from this interview are analysed and the researcher demonstrates the identified problem of the RAE as a result. This is presented in Chapter 5.

2.7.2 Objectives formulation

Objectives formulation as an activity undertaken in this study formulates the objectives for the artefact. In order to formulate these objectives, the descriptive knowledge discussed in Chapter 3 (structured data stores), and the requirements with regard to the situation of the RAE discussed in the first part of Chapter 5, were used. The design objectives for the artefact are presented in Chapter 5 along with problem identification.

2.7.2.1 Suggestions

Suggestions adopted from Vaishnavi and Kuechler's (2004) DSR approach is a sub-activity of the objectives formulation activity. The researcher makes suggestions to support the objectives formulated for the artefact in Chapter 5.

2.7.3 Design and development

The design and development of the artefact is a process constructed from multiple sub-phases similar to that of Vaishnavi and Kuechler (2004).

Within each sub-cycle of the design and development activity, the first phase is to address a problem. This problem is drawn from the objectives for the artefact developed in Chapter 5. For each of these problems, a suggestion is made intuitively to improve the problem to an acceptable level. These suggestions constitute the second phase of the sub-cycle.

The third phase is the development of the solution to the problem according to the stated suggestions. An iterative programming methodology is used whereby the artefact is developed in the first iteration, and modified in each iteration that follows.

2.7.4 Demonstration and evaluation

The demonstration and evaluation activities form part of the evaluation process for this DSR study.

Evaluation is the action of judging materials or methods in terms of internal accuracy and consistency or by comparing them with external criteria (Saunders *et al.*, 2009:591).

The value of the artefact according to Gregor and Hevner (2013:350) is demonstrated using criteria that can include validity, utility, quality and efficacy. The artefact in the context of this study is evaluated on the basis of two criteria selected from Gregor and Hevner (2013). The first criterion is efficacy, and the second criterion is utility. Efficacy is tested during the demonstration activity by using simulated data. Efficacy in this study will relate to acceptability of the artefact according to the objectives developed in Chapter 5. Utility is tested during the evaluation activity by checking if it solves the problem within the RAE. The data gathered from the semi-structured interview with the RAE representative are presented and analysed in Chapter 7, demonstrating the artefact's utility.

2.7.4.1 Communication

The communication activity is a summary of what has been learned from this DSR project. It contributes to the body of knowledge by grouping the lessons learned while the artefact was being developed. The Hevner and Chatterjee (2010:20) checklist is used during the process of the communication activity in Chapter 8, providing a summary of this study.

2.8 CHAPTER CONCLUSION

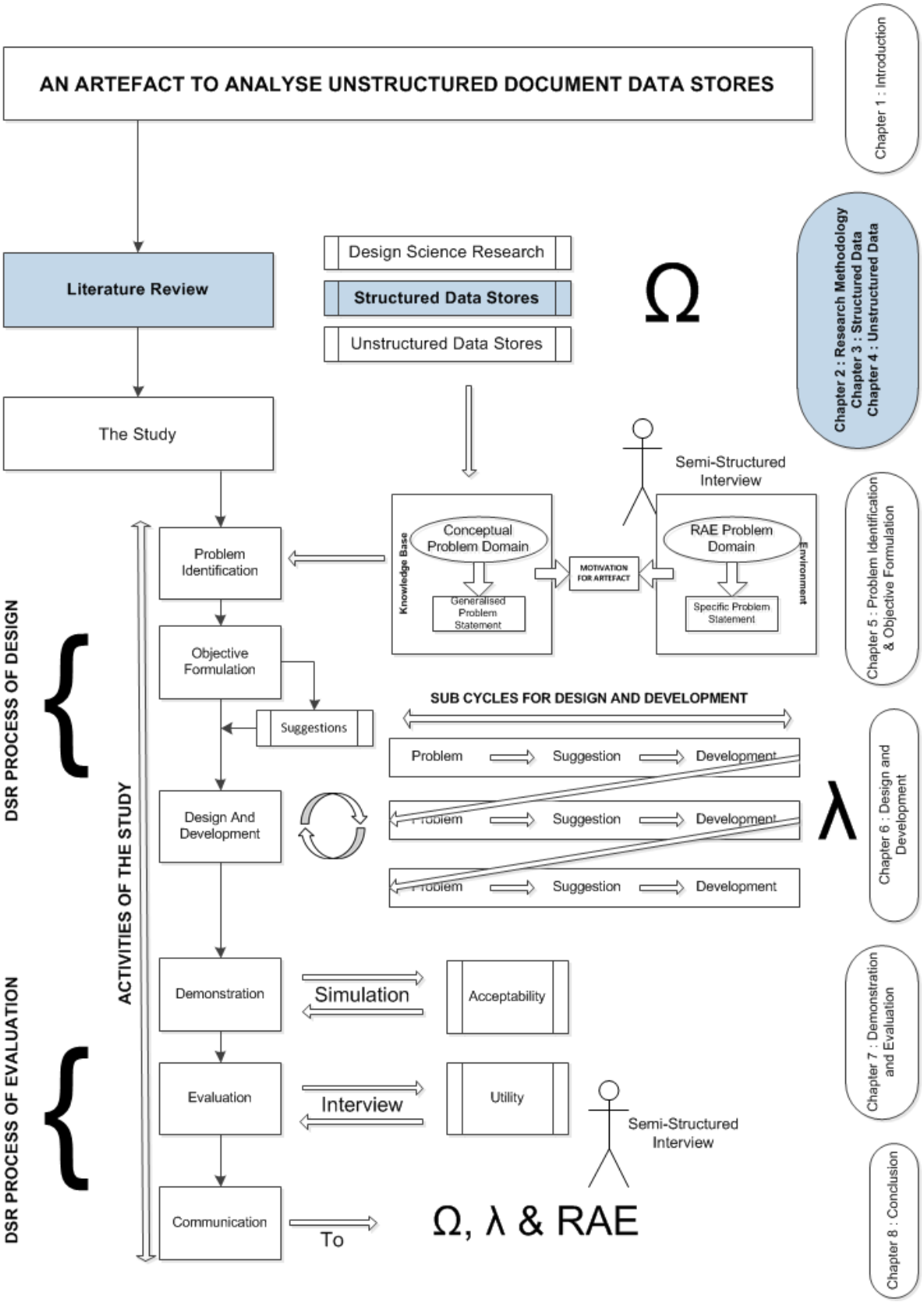
The objective of this chapter was to gain an understanding of research methodology. It focused on the design science research (DSR) paradigm. This objective was met by describing the research, research philosophy, research paradigm, research approach and the process for this study.

DSR methodology was used in this study since the main objective was the creation of an artefact. The process of design, development and evaluation forms a cycle until the artefact is considered useful. The discussion of the KCF and guidelines of DSR provide guidance to the researcher on how to position and evaluate the contribution made by this study.

The research approach of this study explains the processes for designing and evaluating the artefact. These processes are supported by the activities of Peffers *et al.* (2008), combined with the phases of (Vaishnavi & Kuechler, 2004). It is explained how the development of the artefact in this study follows these processes.

The next chapter introduces the literature on structured data stores and contributes to the understanding of the problem and the formulation of the study objectives.

(PAGE INTENTIONALLY LEFT BLANK)


AN ARTEFACT TO ANALYSE UNSTRUCTURED DOCUMENT DATA STORES
Literature Review
The Study
Design Science Research
Structured Data Stores
Unstructured Data Stores
Ω
Semi-Structured Interview
Problem Identification
Objective Formulation
Suggestions
Design And Development
Demonstration
Evaluation
Communication
Knowledge Base
Conceptual Problem Domain
Generalised Problem Statement
MOTIVATION FOR ARTEFACT
RAE Problem Domain
Environment
Specific Problem Statement
SUB CYCLES FOR DESIGN AND DEVELOPMENT
Problem
Suggestion
Development
λ
Simulation
Acceptability
Interview
Utility
Semi-Structured Interview
To
Ω, λ & RAE
DSR PROCESS OF DESIGN
DSR PROCESS OF EVALUATION
ACTIVITIES OF THE STUDY
Chapter 1 : Introduction
Chapter 2 : Research Methodology
Chapter 3 : Structured Data
Chapter 4 : Unstructured Data
Chapter 5 : Problem Identification & Objective Formulation
Chapter 6 : Design and Development
Chapter 7 : Demonstration and Evaluation
Chapter 8 : Conclusion

CHAPTER THREE: STRUCTURED DATA STORES

3.1 INTRODUCTION

The primary objective of this study is to develop an artefact which analyses document data stores in terms of structured data model constructs. In order to achieve this, a discussion of the existing literature on structured data stores is required, which forms part of the descriptive knowledge contributing to this study.

Morris *et al.* (2013:10) define unstructured data as “data that exist in their original (raw) state; that is in the format in which they were collected”, whereas structured data are defined as “unstructured data that have been formatted (structured) to facilitate storage, use, and information generation”. A database is a construct used to facilitate storage, use and information generation of structured data (Morris *et al.*, 2013:10).

A database has two contributory aspects. The first is a collection of raw data, and the second is metadata or data about data in organisations (Morris *et al.*, 2013:7). Typically a database represents the activities of one or more related organisations (Ramakrishnan & Gehrke, 2003:4).

Database management systems (DBMSs) maintain and utilise large collections of data (Ramakrishnan & Gehrke, 2003:4). The DBMS also manages the database design (including metadata) and controls access to the database (Morris *et al.*, 2013:7).

The literature analysis on which this chapter is based begins by discussing the importance of structured data (Section 3.2), followed by an exposition of the evolution of data models and DBMSs (Section 3.3). The relational data model and its characteristics are discussed (Section 3.4), which precedes discussions on

structured query language (Section 3.5). DBMSs are the focus of Section 3.6, followed by the conclusion (Section 3.7).

3.2 IMPORTANCE OF STRUCTURED DATA IN ORGANISATIONS

Modern organisations base their decisions on information (March & Hevner, 2007:1033). The better the information, the better the decisions (Rud, 2009:208; Coronel *et al.*, 2013:5). The data of an organisation may be managed more effectively when the data are stored in a database (Coronel *et al.*, 2013:5). Database design is driven by an understanding of the difference between data and information. Data are raw facts that have not been processed to expose their true value (Coronel *et al.*, 2013:5). Information is the exposed true value of processed data. Hoffer *et al.* (2007:28) explain that "information is data that have been processed in such a way that the knowledge of the person who uses the data increases". Good decision making is grounded in this information (March & Hevner, 2007:1033; Coronel *et al.*, 2013:5).

Coronel *et al.* (2013:5) state that in this information age, the "production of accurate, relevant, and timely information is the key to good decision making". Good decisions guide the organisation towards more knowledge, contributing to the effective management of the organisation (March & Hevner, 2007:1032). Coronel *et al.* (2013:5) define the knowledge age, based on information, as "the body of information and facts about a specific subject". Coronel *et al.* (2013:5) argue that knowledge is the "familiarity, awareness and understanding of information as it applies to an environment".

Information requires accurate data that must be generated and stored properly. This is facilitated by a discipline known as data management. According to Coronel *et al.* (2013:5) and Morris *et al.* (2013:6), data management is focused on the proper generation, storage and retrieval of data. The data obtained and the management thereof are therefore the core activities of any organisation.

The transformation of data into information and information into knowledge within an organisation's context is constructed within a framework labelled business intelligence (BI). March and Hevner (2007:1032) state that the term 'business intelligence' "refers to inferences and knowledge discovered by applying algorithmic analysis to acquired information". Morris *et al.* (2013:732) provide a more comprehensive description of BI as "a term used to describe a comprehensive, cohesive, and integrated set of tools and processes used to capture, collect, integrate, store, and analyse data with the purpose of generating and presenting information used to support business decision making". BI is drawn from an intelligence repository known as a data warehouse (March & Hevner, 2007:1032). Data warehouses store structured data – Inmon (2005:31) describes a data warehouse as a "subject-oriented, integrated, time-invariant, non-updatable collection of data used to support management decision-making processes and business intelligence". BI improves a business's performance through active decision making which empowers the user to make good decisions based on collected knowledge of the business (Morris *et al.*, 2013:733). BI, according to Rud (2009:3), "encompasses all the capabilities required to turn data into intelligence, [and] has emboldened companies to strive for the ultimate goal: getting the right information to the right people at the right time through the right channel". As Morris *et al.* (2013:734) state:

> *"BI is not a product by itself, but a framework of concepts, practices, tools, and technologies that help a business better understand its core capabilities, provide snapshots of the company situation, and identify key opportunities to create competitive advantage"*.

The steps involved in developing BI are outlined in Table 3.1.

Table 3.1: Steps involved in developing BI quoted from Morris et al. (2013:637)

STEP	DESCRIPTION
1	Operational data collection and storage.
2	Decision support data generation by aggregating operational data.
3	Generate information by analysing decision support data.
4	To support business decisions by presenting the information to the end user.
5	Making business decisions, which in turn generate more data that is collected, stored, etc.
6	Evaluate the outcomes of business decisions by monitoring results.
7	A high degree of accuracy into future predictions of business behaviour.

Kimball *et al.* (2008:597) state that "Though some would argue that you can theoretically deliver BI without a data warehouse, and vice versa, that is ill-advised from our perspective". There are different data models to organise data in a data warehouse, but all data models use basic constructs of relational databases to store data. The management of data, information and BI are facilitated by well-designed relational databases, data models, and DBMSs (Rob *et al.*, 2008:10; Coronel *et al.*, 2013:13; Morris *et al.*, 2013:11). These constructs of data models and DBMSs are discussed in the following sections.

3.3 EVOLUTION OF DATA MODELS AND DBMS

The concept of databases has evolved since the 1960s, aiding the development, construction and management of information systems. Until the 1970s, data were not particularly flexible and were difficult to navigate. Developers needed to know exactly what customers wanted to achieve with the data before databases were designed (Coronel *et al.*, 2013:3). Data models and database management systems are key aspects in the development of databases.

Data models are simple graphical representations of complex real-world database structures (Morris *et al.*, 2013:75). Data models are also representations of organisational data or part of an organisation's data requirements (Connolly *et al.*, 2008). Kimball and Ross (2011:10) state that "a data model that starts by being simple has a chance of remaining simple at the end of the design".

Morris *et al.* (2013:76) state that "the basic building blocks for data models are entities, attributes, relationships and constraints". An entity is a collection of data about any phenomenon or object. Entities can represent physical objects, such as people, cars and products, or abstract objects, such as schedules, enrolments or events. Attributes are characteristics of an entity and define the properties of an entity. Relationships describe the associations found between entities. Relationships consist of three distinguishable types: one-to-one, one-to-many and many-to-many. Each of these types of relationships are explained in the discussion of the relational data model in Section 3.4.1.5. The final building block for data models is the constraints that are placed on the data. A constraint places a restriction on the data and helps to ensure data integrity. These building blocks of data models are discussed in more detail in Section 3.4.1.

Data models are implemented by a DBMS to facilitate the management of data. A DBMS maintains and utilises large collections of data (Ramakrishnan & Gehrke, 2003:4). The DBMS also manages the database design and controls access to the database (Morris *et al.*, 2013:7).

The DBMS concept has been in existence since the 1960s. During that time, the first general-purpose DBMS, known as the integrated data store, was designed at General Electric by Charles Bachman (Ramakrishnan & Gehrke, 2003:6). Bachman's DBMS development strongly influenced DBMSs progress when the Conference on Data System Languages (CODASYL) standardised the DBMS progress. The network data model's (discussed in Section 3.3.1) foundations were based on Bachman's DBMS development. At the same time, an Information Management System (IMS) was developed by International Business Machines

(IBM), which formed an alternate data representation structure. This structure was named the hierarchical data model (discussed in Section 3.3.2). The IMS led to the Semi-Automated Business Research Environment (SABRE), an airline reservation system co-developed by American Airlines (AA) and IBM (Ramakrishnan & Gehrke, 2003:6). SABRE was one of the first systems that allowed several users to access data concurrently (Ramakrishnan & Gehrke, 2003:6).

A new data representation structure (model) was proposed in 1970 by Dr E. F. Codd, known as the relational data model, which is discussed in Section 3.4. This new data model formed a solid foundation for the development of several DBMSs based on the relational model (Ramakrishnan & Gehrke, 2003:6).

The relational model has become the dominant DBMS since the 1980s, when it acquired widespread acceptance. During the late 1980s and early 1990s, relational database systems were extended by various vendors to support more complex data types such as images and text (Ramakrishnan & Gehrke, 2003:6).

Various other data models have emerged since the 1980s, including the Object-Oriented Model (OOM) and the Extended Relational Data Model (ERDM). In 2009, the most significant data model or models that emerged were NoSQL data models. NoSQL as an example of unstructured data models is discussed in Chapter 4.

In the following sections the network model and the hierarchical model are briefly described as they relate closely to the relational model. An in-depth description of the relational data model is presented in Section 3.4.

3.3.1 The network model

The network model represents a collection of one-to-many relationships (discussed in Section 3.4.1.5.2) between data records. The network model was created to represent complex data relationships more effectively when compared to the

hierarchical model. The network model improved database performance and facilitated the development of database standards (Morris *et al.*, 2013:80).

At this early stage no standards existed and programmers and software designers found that the lack of database standards were troublesome, making the database design and application less portable. In the late 1960s, the Conference on Data System Languages (CODASYL), created a group called the Database Task Group (DBTG) to establish database standards. The DBTG compiled a standard set of specifications. These specifications would encourage database development and data manipulation in a given environment. The final report summarised all the important database components: a network schema, a network sub-schema, a data management language (DML), and a data definition language (DDL) (Morris *et al.*, 2013:81). These database components are still implemented in data models today.

Figure 3.1 illustrates an example of the network model which represents a hotel room bookings system.

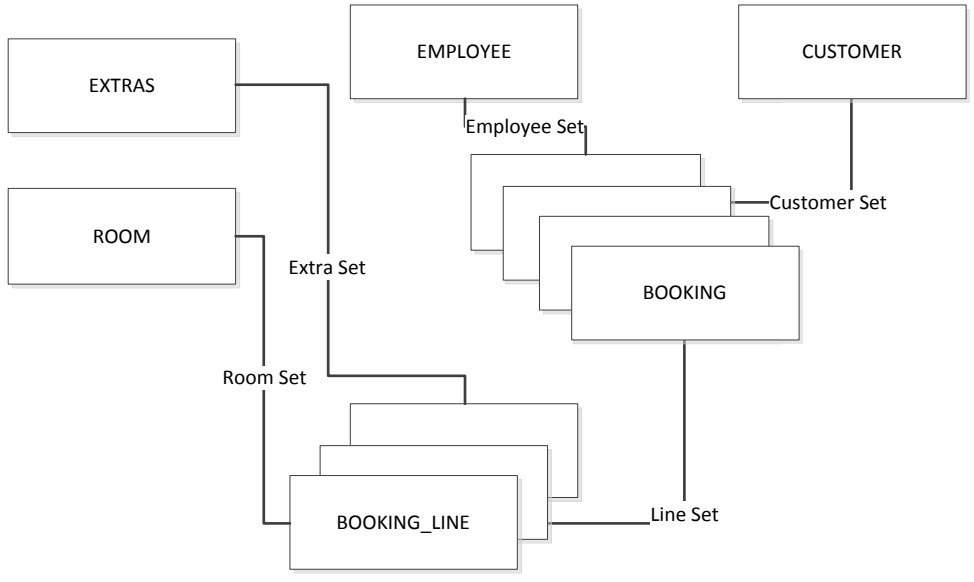


Figure 3.1: Example of a network model data representation

Each customer can make many bookings. Bookings are handled by employees. A booking can be for more than one room. Rooms also have extras such as a minibar, hot tub and internet access. In Figure 3.1, the “BOOKING_LINE” entity contains various parent entities (“BOOKING”, “ROOM”, “EXTRAS”) and the “BOOKING” entity also contains various parent entities (“EMPLOYEE”, “CUSTOMER”).

3.3.2 The hierarchical model

The hierarchical model can be represented using an upside-down tree diagram. It contains levels and segments. A segment in the hierarchical model is representative of an entity known in the 1960s as a file systems record. A number of one-to-many relationships (discussed in Section 3.4.1.5.2) may exist between a parent and child segment. The hierarchical model had limitations: it was complex to implement; it yielded structural independence; and it was difficult to manage (Rob *et al.*, 2008:38).

Figure 3.2 is an example of the hierarchical model. A car is an entity composed of various sections such as structure, door assembly and engine assembly. Each of these sections is composed of sub-sections such as the structure, which includes a paint layer and a bumper. As such, every component is drilled down, forming a tree structure as in Figure 3.2. In the figure, ‘Car’ is the root segment of ‘Structure’, ‘Door Assembly’ and ‘Engine Assembly” are child segments. These three segments then become the root and parent segments for each of their child segments.

The hierarchical model is similar to the network model regarding many aspects. The hierarchical model represents one-to-many relationships with only one parent for each child, but the network model differs from the hierarchical model in the way that a child may have more than one parent.

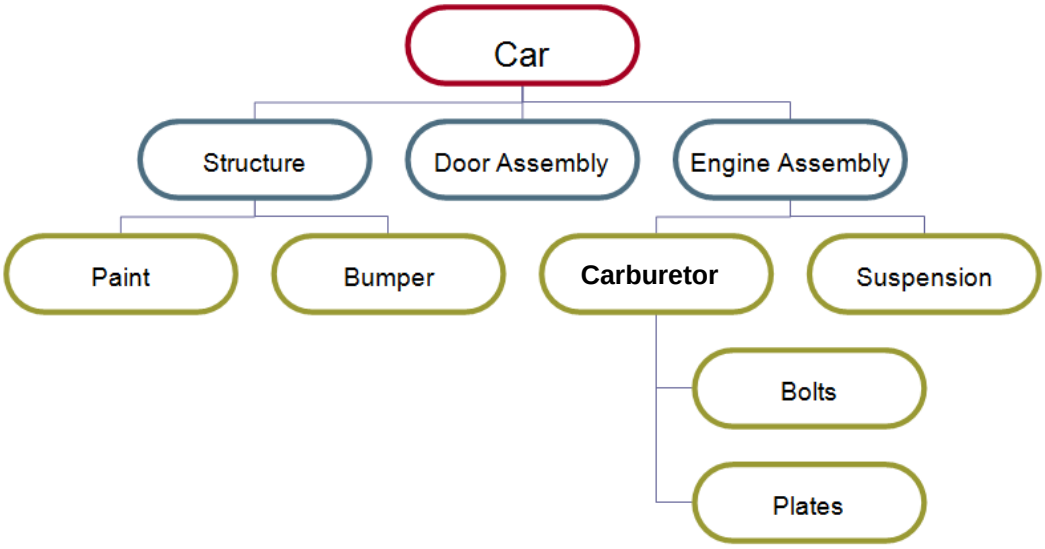


Figure 3.2: Example of a hierarchical model data representation

3.4 THE RELATIONAL MODEL

Dr E. F. Codd introduced the relational model in his landmark seminar paper "A relational model of data for large shared data banks" (Codd, 1970). This model created a major eruption in the fields of computing and is known to have spurred a genuine database revolution (Coronel *et al.*, 2013:41). At the time, the relational model was considered ingenious but impractical, due to the required computing power that was not available during those years (Coronel *et al.*, 2013:41).

The mathematical concept, known as a relation, is the foundation of the relational model. A relation may be thought of as a matrix, composed of intersecting columns and rows, thereby avoiding the complexity of abstract mathematical theory. A relation is referred to as a table, in which a row consists of a tuple and a column consists of an attribute (Ramakrishnan & Gehrke, 2003).

The relational database management system (RDBMS) implements the relational model (Morris *et al.*, 2013:81). RDBMSs support similar functions to those of the network and hierarchical DBMSs, but expand functionality with regard to the

understanding and implementation of the relational model. The complexities of the relational model, like the physical details, are hidden from the user in the RDBMS. The user only sees a collection of tables, which are related and may be manipulated and queried. Each table is a matrix and they are related to one another by a common attribute.

The two tables in Figure 3.3 provide an example of data representation in a relational model. The first table “CUSTOMER” is related to the second table “INVOICE” by the common attribute “CUSTOMER_ID”. By matching the records, the user is able to see what purchases have been made by each customer. Although these tables are independent of each other, they can easily be associated by the user.

Table name: CUSTOMER

CUSTOMER_ID	NAME	SURNAME	ID_NUMBER	PHONE
CUS101101	Kelly	Mac Pearson	8802021243056	0169521122
CUS101201	Justin	Mac Pearson	8412155102057	0111245778
CUS106102	Alice	Creswell	5508250012075	0112124545
CUS203101	Alex	Seamen	7501021002085	0211216532

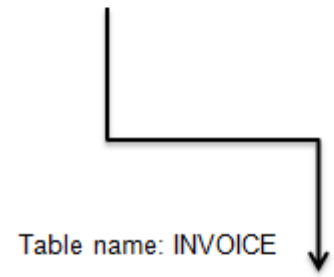


Table name: INVOICE

INVOICE_NUM	CUSTOMER_ID	DATE	TOTAL
INV0012	CUS101101	2012-01-02	417-00
INV0013	CUS101201	2012-02-29	25-00
INV0014	CUS203101	2013-05-25	142-00
INV0032	CUS101201	2013-08-02	925-00
INV0035	CUS101201	2013-10-16	2923-00
INV0065	CUS101101	2013-11-16	545-00
INV0066	CUS106102	2013-11-17	1925-00

Figure 3.3: Example data representation of the relational model

The relational model is at times represented by the relational diagram. The relational diagram is a simple representation of tables, attributes within tables, and relationships between tables. Figure 3.4 is a relational diagram depiction of the model illustrated in Figure 3.3.

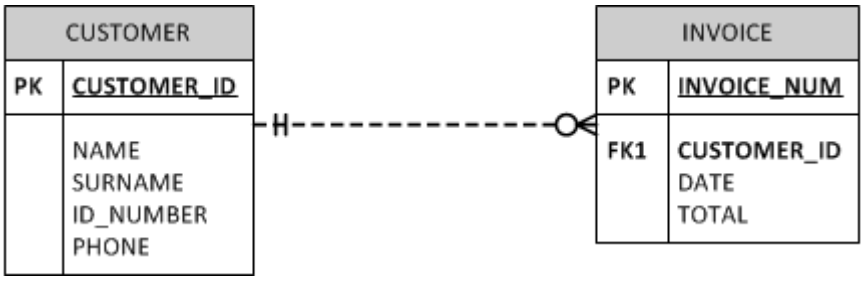


Figure 3.4: Relational diagram between "CUSTOMER" and "INVOICE"

Tables within the relational model are also referred to as entities. An entity is a logical construct and contains a collection of related entity occurrences. A collection of entity occurrences, stored in a relational table, resembles a file. Coronel *et al.* (2013:72) stress the logical nature of the model: "A table yields complete data and structural independence because it is a purely logical structure", and "How the data is stored in the database is of no concern to the end user or the designer". In essence the relational model should be viewed as a logical structure.

The implementation of a powerful and flexible query language, Structured Query Language (SQL), gave the relational model another reason to rise to authority. This powerful and flexible query language was built on relational algebra that was also introduced by Dr E. F. Codd (1983). SQL is based on relational algebra. SQL is found within most RDBMSs. SQL is explored in Section 3.5.

The characteristics of the relational model are explored next.

3.4.1 Relational model characteristics

Since the relational model may be view as a logical structure, it contains characteristics. These characteristics include: a logical view, keys, integrity rules, data dictionary, system catalogue and relationships. Figure 3.3 serves as an example to demonstrate most of these characteristics.

3.4.1.1 Data in a logical view

A table is a logical construct that facilitates the creation of data relationships. This logical table is known as a logical view in the relational database. A table can store entity occurrences or related entities also known as an entity set. The attributes that are found within these tables have a certain set of values, known as a domain. Tables also contain a degree and cardinality. The degree refers to the number of attributes in the table and the cardinality refers to the number of rows in a table (Connolly & Begg, 2010:96; Coronel *et al*., 2013:75). Figure 3.5 illustrates the degree and cardinality of the CUSTOMER table in Figure 3.3.

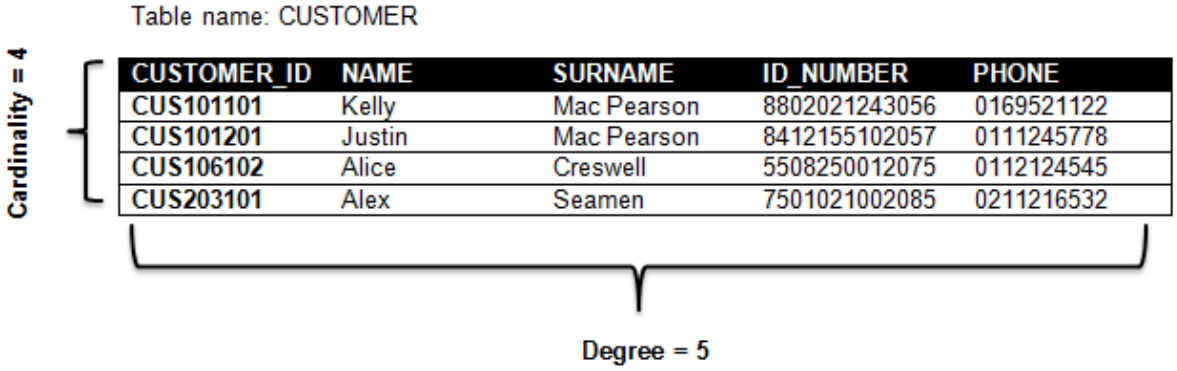


Figure 3.5: Demonstration of degree and cardinality for "CUSTOMER"

Tables also have a textual representation. This representation is known as a relational schema. A relational schema lists the table's name and is followed by a list of attributes that are placed in parentheses. Figure 3.6 illustrates the relational schema representation of the "CUSTOMER" and the "INVOICE" table in Figure 3.3.

CUSTOMER (CUSTOMER_ID, NAME, SURNAME, ID_NUMBER, PHONE)

INVOICE(INVOICE_NUM, CUSTOMER_ID, DATE, TOTAL)

Figure 3.6: Demonstration of relational schema for "CUSTOMER" and "INVOICE"

Table 3.2 lists the characteristics of a relational table.

Table 3.2: Characteristics of a relational table quoted from Coronel et al. (2013:73) and Morris et al. (2013:106)

	DESCRIPTION
1	A table is perceived as a two-dimensional structure composed of rows and columns.
2	Each table row (tuple) represents a single entity occurrence within the entity set.
3	Each table column represents an attribute, and each column has a distinct name.
4	Each row/column intersection represents a single data value.
5	All values in a column must conform to the same data format.
6	Each column has a specific range of values known as the attribute domain.
7	The order of the rows and columns is immaterial to the DBMS.
8	Each table must have an attribute or a combination of attributes that uniquely identifies each row.

3.4.1.2 Keys

In the relational data model, keys play a crucial role. They make each row within a table uniquely identifiable. A key may consist of one or more attributes that determine other attributes. Keys are also used to build relationships between tables and ensure the integrity of data. There are several types of relational database keys which are summarised in Table 3.3.

Table 3.3: Relational database keys quoted from Coronel et al. (2013:83) and Morris et al. (2013:111)

KEY TYPE	DEFINITION
Super key	An attribute (or combination of attributes) that uniquely identifies each row in a table.
Candidate key	A minimal (irreducible) super key. A super key that does not contain a subset of attributes.
Primary key	A candidate key selected to uniquely identify all other attribute values in any given row. Cannot contain null entries.
Secondary key	An attribute (or combination of attributes) used strictly for data retrieval purposes.
Foreign key	An attribute (or combination of attributes) in one table, whose values must either match the primary key in another table or be null.

The highlighted “CUSTOMER_ID” attribute serves as a primary key (PK) for the “CUSTOMER” table in Figure 3.7 and the “ID_NUMBER” attribute serves as a candidate key (CK) or secondary key (SK).

Table name: CUSTOMER

CUSTOMER_ID	NAME	SURNAME	ID_NUMBER	PHONE
CUS101101	Kelly	Mac Pearson	8802021243056	0169521122
CUS101201	Justin	Mac Pearson	8412155102057	0111245778
CUS106102	Alice	Creswell	5508250012075	0112124545
CUS203101	Alex	Seamen	7501021002085	0211216532

PK CK/SK

Figure 3.7: Demonstration of primary key and candidate key

The “INVOICE_ID” attribute serves as a primary key (PK) for the “INVOICE” table in Figure 3.8 and the “CUSTOMER_ID” attribute serves as a foreign key (FK). Both “INVOICE_ID” and “CUSTOMER_ID” could serve as a super key (SPK).

Table name: INVOICE

INVOICE_NUM	CUSTOMER_ID	DATE	TOTAL
INV0012	CUS101101	2012-01-02	417-00
INV0013	CUS101201	2012-02-29	25-00
INV0014	CUS203101	2013-05-25	142-00
INV0032	CUS101201	2013-08-02	925-00
INV0035	CUS101201	2013-10-16	2923-00
INV0065	CUS101101	2013-11-16	545-00
INV0066	CUS106102	2013-11-17	1925-00

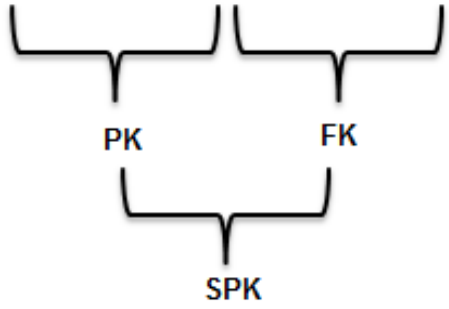


Figure 3.8: Demonstration of primary key, foreign key and super key

3.4.1.3 Integrity rules

Good database design relies on integrity rules. The integrity rules are automatically enforced by the RDBMS, but it is recommended that the application design conforms to these rules. These rules are summarised in Table 3.4.

Table 3.4: Relational database keys adapted from Coronel et al. (2013:84) and Morris et al. (2013:112)

ENTITY INTEGRITY	DESCRIPTION
Requirement	• All entries of primary keys are unique, and • No part of a primary key may be null.
Purpose	• Each row will have a unique identity.

	• Foreign key values can properly reference primary key values.
Example	In the example of Figure 3.3 no invoice may contain a duplicate number or may be null.
REFERENCE ENTEGRITY	**DESCRIPTION**
Requirement	• If a foreign key it is not a part of its table's primary key it may have a null entry. • An existing primary key value must be reference for a non-null foreign key value.
Purpose	• It is possible for a corresponding value not to be related, thereby restricting invalid entries. • It imposes the restriction that a record may not be deleted if that record is a mandatory foreign key in another table.
Example	In Figure 3.3 an invoice may for example not have contained a customer number, since the invoice may not have been assigned to a customer yet.

3.4.1.4 Data dictionary and system catalogue

All tables found within a relational database contain administrative details such as a list of attributes and characteristics. These descriptions of the tables are kept in the data dictionary. The data dictionary therefore contains metadata. Metadata are widely referred to as 'data about data', meaning that it is data that describes other data. At times, the data dictionary is seen as the database designers' database (Coronel *et al.*, 2013:87).

The system catalogue is similar to the data dictionary as it also contains metadata. Coronel *et al.* (2013:87) state that the system catalogue is a detailed data dictionary and all objects in the database are described in the system catalogue. According to Coronel *et al.* (2013:87), these objects include "data about table names, the table's creator and creation date, the number of columns in each table, the data type corresponding to each column, index filenames, index creators, authorised users and access privileges".

Database documentation is automatically provided by the system catalogue. Figure 3.9 is a representation of the system catalogue for Figure 3.3.

Table Name	Attribute Name	Type	Format	Domain	REQUIRED	PK or FK	FK Reference
CUSTOMER	CUSTOMER_ID	VCHAR	CUSXXXXX	0-999999999	Y	PK	INVOICE
	NAME	VCHAR2		Xxxxxx			
	SURNAME	VCHAR2		Xxxxxx			
	ID_NUMBER	VCHAR		XXXXXXXXX			
	PHONE	CHAR		XXXXXXXXX			
INVOICE	INVOICE_NUM	INT	INVXXXXX	0-999999999	Y	PK	
	CUSTOMER_ID	VCHAR	CUSXXXXX	0-999999999	Y	FK	
	DATE	DATE	yyyy-mm-dd				
	TOTAL	DECIMAL		0-999999999			

Figure 3.9: System catalogue of "CUSTOMER" and "INVOICE"

3.4.1.5 Relationships

Relationships are central to the relational model. Relationships are classified as one-to-one (1:1), one-to-many (1:M) and many-to-many (M:N); these three types of relationships are discussed in the following sections.

3.4.1.5.1 One-to-One relationships

The one-to-one relationship involves one entity instance that may only be related to one other entity instance and vice versa (Morris *et al.*, 2013:123). For example: a customer of a hardware store may place a number of purchase orders for numerous items on numerous occasions. For each purchase only one invoice may be generated, and only this invoice is assigned to that order. An invoice may not be generated without a purchase order, but a purchase order may still exist or be created without an invoice. This creates a one-to-one relationship between the purchase order and the invoice. This implies that one-to-one relationships are single-valued in both directions. One-to-one relationships are rarely found in relational database design. However, within certain conditions such as a generalisation hierarchy, one-to-one relationships are an absolute necessity (Morris *et al.*, 2013:123). Generalisation hierarchy does not fall within the scope of this study and is

not discussed and explained. Figure 3.10 is a representation of a one-to-one relationship, and in this instance one order is related to only one invoice.

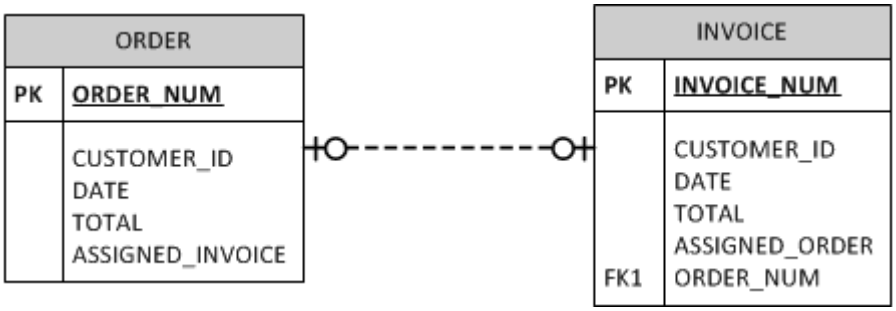


Figure 3.10: One-to-one relationship between "ORDER" and "INVOICE"

Figure 3.11 is a data representation of Figure 3.10. In this example, it can be seen that the "ORDER" is related to only one "INVOICE" and an "INVOICE" is related to only one "ORDER".

3.4.1.5.2 One-to-Many relationships

The one-to-many relationship implies that one entity instance can have multiple related instances within another entity. For example: a customer of a hardware store may purchase numerous items on numerous occasions. An invoice is generated for each purchase on each of these occasions. This creates a one-to-many relationship between the customer and the generated multiple invoices. One-to-many relationships are the norm for relational databases (Morris *et al.*, 2013:121). Figure 3.12 is a representation of a one-to-many relationship. In the figure, the "CUSTOMER" table (one) is related to many invoice occurrences in the "INVOICE" table (many). Figure 3.3 serves as a data example for this type of relationship.

Table name: ORDER

ORDER_NUM	CUSTOMER_ID	DATE	TOTAL	ASSIGNED_INVOICE
ODN0012	CUS101101	2012-01-02	417-00	INV0012
ODN0013	CUS101201	2012-02-29	25-00	INV0013
ODN0017	CUS203101	2013-05-25	142-00	INV0014
ODN0082	CUS101201	2013-08-02	925-00	
ODN0135	CUS101201	2013-10-16	2923-00	INV0035
ODN0265	CUS101101	2013-11-16	545-00	INV0065
ODN0466	CUS106102	2013-11-17	1925-00	INV0066

Table name: INVOICE

INVOICE_NUM	CUSTOMER_ID	DATE	TOTAL	ASSIGNED_ORDER
INV0012	CUS101101	2012-01-02	417-00	ODN0012
INV0013	CUS101201	2012-02-29	25-00	ODN0013
INV0014	CUS203101	2013-05-25	142-00	ODN0017
INV0035	CUS101201	2013-10-16	2923-00	ODN0135
INV0065	CUS101101	2013-11-16	545-00	ODN0265
INV0066	CUS106102	2013-11-17	1925-00	ODN0466

Figure 3.11: Example data representation of a one-to-one relationship between "ORDER" and "INVOICE"

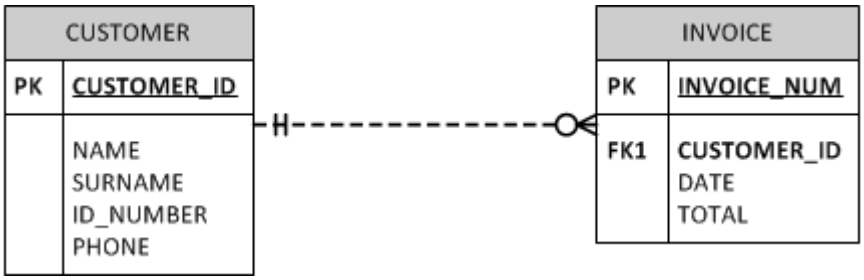


Figure 3.12: One-to-many relationship between "CUSTOMER" and "INVOICE"

3.4.1.5.3 Many-to-Many relationships

The many-to-many relationship implies that multiple entities can have related instances within other entities. The many-to-many relationship is not directly supported in the relational database environment. These relationships are resolved by creating an associative entity, referred to as a bridge or a composite entity (Morris

et al., 2013:127). The associative entity uses one-to-many relationships between the entities. Figure 3.13 is a demonstration of a many-to-many relationship. In Figure 3.13, an invoice (“INVOICE”) may contain many products (“PRODUCT”), and a product may be used in many invoices. Figure 3.14 is a representation of the solved many-to-many relationship illustrated in Figure 3.13 by implementing an associative entity (“INVOICE_LINE”) and one-to-many relationships.

Figure 3.13: Many-to-many relationship between “INVOICE” and “PRODUCT”

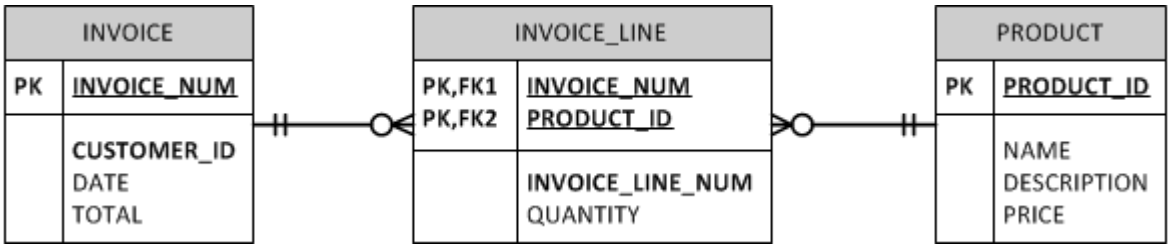


Figure 3.14: One-to-many relationship between “INVOICE”, “INVOICE_LINE” and “PRODUCT”

Figure 3.15 is a data representation of Figure 3.14. In this example it can be seen that one “INVOICE” contains many “INVOICE_LINE” records: this forms a one-to-many relationship between “INVOICE” and “INVOICE_LINE”. From the example it can also be seen that many “PRODUCT” records are contained in many “INVOICE_LINE” records: this forms a one-to-many relationship between “PRODUCT” and “INVOICE_LINE”.

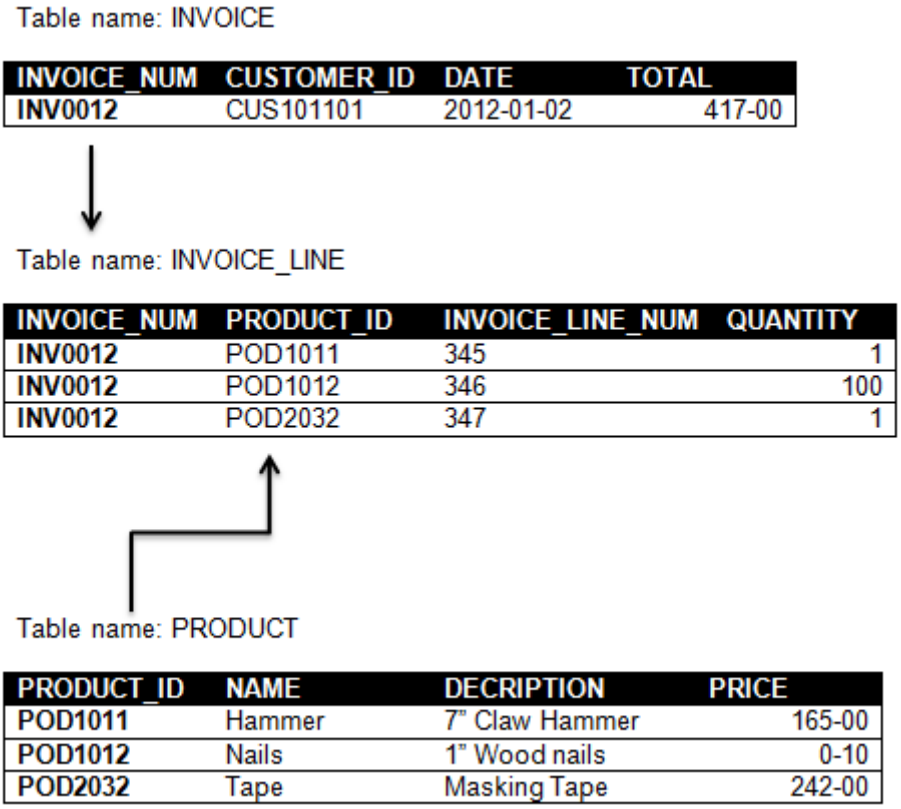


Figure 3.15: Example data representation of one-to-many relationship between "INVOICE", "INVOICE_LINE" and "PRODUCT"

3.4.2 Conclusion of the relational data model

The relational model is a representation of related data. It is made up of entities that contain attributes and the relationships found between entities. A list of 12 rules was published by Codd (1985a, 1985b). This list was intended to define a relational database system. The reason for the publication was because many vendors marketed their products as relational, but these products did not meet the minimum standards. Table 3.5 lists the 12 rules of Dr E. F. Codd.

Table 3.5: Codd's 12 relational database rules quoted from (Codd, 1985b)) and (Codd, 1985a)[3]

RULE	RULE NAME	DESCRIPTION
1	**Information**	All information in a relational database must be logically represented as column values in rows within tables.
2	**Guaranteed Access**	Every value in a table is guaranteed to be accessible through a combination of table name, primary key value, and column name.
3	**Systematic Treatment of Nulls**	Nulls must be represented and treated in a systematic way, independent of data type.
4	**Dynamic Online Catalog Based on the Relational Model**	The metadata must be stored and managed as ordinary data, that is, in tables within the database. Such data must be available to authorised users using the standard database relational language.
5	**Comprehensive Data Sublanguage**	The relational database may support many languages. However, it must support one well-defined, declarative language with support for data definition, view definition, data manipulation (interactive and by program), integrity constraints, authorisation and transaction management (begin, commit and roll back).
6	**View Updating**	Any view that is theoretically updatable must be updatable through the system.
7	**High-Level Insert, Update, and Delete**	The database must support set-level inserts, updates and deletes.
8	**Physical Data Independence**	Application programs and ad hoc facilities are logically unaffected when physical access methods or storage structures are changed.
9	**Logical Data Independence**	Application programs and ad hoc facilities are logically unaffected when changes are made to the table structures that preserve the original table values (changing order of columns or inserting columns).

[3] The original page number of the articles could not be determined as the available digital format of these articles does not reflect the original page numbers.

10	**Integrity Independence**	All relational integrity constraints must be definable in the relational language and stored in the system catalog, not at the application level.
11	**Distribution Independence**	The end users and application programs are unaware and unaffected by the data location (distributed vs. local databases).
12	**Nonsubversion**	If the system supports low-level access to the data, there must not be a way to bypass the integrity rules of the database.
	Rule Zero	All preceding rules are based on the notion that in order for a database to be considered relational, it must use its relational facilities exclusively to manage the database.

The following section continues with the discussion of SQL.

3.5 STRUCTURED QUERY LANGUAGE

Structured Query Language (SQL) is a well-known programming language used by RDBMSs to manage and manipulate data. SQL was developed for the original RDBMS to form the interaction interface between the user and the data (Connolly & Begg, 2005:70; Connolly & Begg, 2010:136; Morris *et al.*, 2013:83). The SQL language was first standardised in 1986 and then improved on in 1989. A significant standard version of the SQL was completed in 1992. The RDBMSs provide access to relational databases using SQL. SQL forms part of the database access languages and application programming interface functions of DBMSs (Coronel *et al.*, 2013:27).

SQL is used by the RDBMS to interpret user enquiries and transforms them into commands to be executed. The use of SQL makes data retrieval effortless. Three parts are involved in any SQL-based relational database application. These three parts are: an end-user interface, a set of tables stored in the database, and a SQL engine. The three parts are discussed in Table 3.6.

Table 3.6: Three parts of a database application summarised from Coronel et al. (2013:43).

PART	DESCRIPTION
The end-user interface	• The interface allows the end user to interact with the data (by auto-generating SQL code). • The design of the interface is a product of the software vendor's idea of meaningful interaction with the application. • The design of the interface may also be a product of design of the developers' or users' own customised interface.
A collection of tables stored in the database	• Data are stored in tables. • Tables simply 'present' the data to the end user. • Tables are easy to understand. • Each table is independent. • Rows in different tables are related by common values in common attributes.
SQL Engine	• Largely hidden from the end user. • SQL engine executes all queries, or data requests. • Forms part of the DBMS software. • Uses SQL to create table structures. • Uses SQL to perform data access. • Uses SQL for table maintenance. • Processes all user requests. • SQL is a declarative language.

The SQL language consists of two categories: a data definition language (DDL) and a data manipulation language (DML) (Coronel *et al.*, 2013:313). Table 3.7 describes these two languages. Table 3.8 lists the DDL commands and Table 3.9 lists the DML commands.

Table 3.7: Description of the data definition language (DDL) and data manipulation language (DML) summarised from Coronel et al. (2013:313).

CATEGORY	DESCRIPTION
Data Definition Language (DDL)	Commands to create database objects such as tables, indexes and views, as well as commands to define access rights to those database objects.
Data Manipulation Language (DML)	Commands to insert, update, delete, and retrieve data within the database tables.

Table 3.8: Data definition language (DDL) adapted from Coronel et al. (2013:313).

COMMAND OR OPTION	DESCRIPTIONS
CREATE SCHEMA AUTHORISATION	Creation of a new database schema.
CREATE TABLE	Creation of a new table in the database schema.
NOT NULL	Prevents null values within a column.
UNIQUE	Prevents duplicate values within a column.
PRIMARY KEY	For a table it specifies the primary key.
FOREIGN KEY	For a table it specifies the foreign key.
DEFAULT	For a column (when no value is given) it specifies the default value.
CHECK	Attribute data are validated.
CREATE INDEX	A table index is created.
CREATE VIEW	Creates a dynamic subset of rows/columns from one or more tables.
ALTER TABLE	Table definitions are modified (adds, modifies or deletes attributes or constraints).

CREATE TABLE AS	Creates a new table based on a query in the user's database schema.
DROP TABLE	Removes table and data permanently.
DROP INDEX	Removes index permanently.
DROP VIEW	Removes view permanently.

Table 3.9: Data manipulation language (DML) adapted from Coronel et al. (2013:43).

COMMAND OR OPTION	DESCRIPTIONS
INSERT	Inserts a record.
SELECT	Selects attributes from rows in one or more tables or views.
WHERE	Limits the returned data based on criteria.
GROUP BY	Using one or more attributes to group the rows returned.
HAVING	Using a condition to limit the selection of grouped rows.
ORDER BY	Using one or more attributes to sort the order of the returned rows.
UPDATE	Modifies data of the attributes.
DELETE	Removes records from the table.
COMMIT	Changes data permanently.
ROLLBACK	Original values of data are restored.
Comparison Operators	
=, <, >, <=, >=, <>	Used for comparison purposes.
Logical Operators	
AND/OR/NOT	Used for comparison purposes.
Special Operators	Used for comparison purposes.

BETWEEN	Compares the attribute to a range of values.
IS NULL	Compares the attribute to the null value.
LIKE	Compares the attribute to a string pattern.
IN	Compares the attribute to any matched value within a list.
EXIST	Determines whether any rows are return by a subquery.
DISTINCT	Limits values to unique values.
Aggregate Functions	Mathematical function available for a SELECT statement.
COUNT	For a given column the number of rows with non-null values is returned.
MIN	For a given column the highest value for an attribute is returned.
MAX	For a given column the smallest value for an attribute is returned.
SUM	For a given column the total of all values is returned.
AVG	For a given column the average of all values is returned.

SQL is relatively easy to learn, but further discussion thereof is beyond the scope of this study. RDBMSs are briefly discussed next.

3.6 RELATIONAL DATABASE MANAGEMENT SYSTEM

The Relational DBMS (RDBMS) is defined as a database management system that maintains data records and indexes in tables. Within the tables, data relationships may be created and maintained (Connolly *et al.*, 2008; Agrawal, 2012; Morris *et al.*, 2013:81).

An in-depth discussion of RDBMSs falls beyond the scope of this study and a brief description of atomicity, consistency, isolation and durability (ACID) properties, functions, advantages and disadvantages are listed.

3.6.1 Atomicity, Consistency, Isolation and Durability

A very specific aspect found in RDBMSs is that they provide atomicity, consistency, isolation and durability (ACID) properties for transactions. ACID properties ensure consistency between database instances in the event of concurrent access and potential system failures. This is accomplished by a technique known as two-phase commit protocol (2PC) (Pritchett, 2008:50). The properties of ACID are defined as follows:

- Atomicity means that either all operations in the transaction will complete, or else none will (Pritchett, 2008:50; Morris *et al.*, 2013:573). To elaborate further, given a transaction with a set of different operations to be executed, should any one of these operations fail, then the entire transaction should fail. This means that changes will be rolled back to bring the database to an unchanged state (Oliveira da Silva, 2011:7).
- Consistency refers to the fact that a database will not violate any integrity constraints. The database will be in a consistent state when the transaction begins or ends (Pritchett, 2008:50; Morris *et al.*, 2013:573). It insures that only valid data are written to the database. Thus the database will be taken from one consistent state to a new consistent state, but only if the transaction has been executed successfully. Should any errors arise in the process of a transaction, then it is to be rolled back to the previous state (Oliveira da Silva, 2011:7).
- Isolation refers to the fact that a transaction will distinguish itself as the only operation being performed within a database (Pritchett, 2008:50; Morris *et al.*, 2013:573). An executed transaction may not impact any other concurrent transactions being executed; it is to remain isolated while it is not yet committed. If a transaction is not isolated, it may result in the access of inconsistent data from the system (Oliveira da Silva, 2011:7).

- Durability guarantees that once a transaction is completed and committed, that transaction cannot be reversed (Pritchett, 2008:50; Morris *et al.*, 2013:573). Committed transactions are permanent and cannot be lost. In case of any hardware or software-related failure within the system, the database must be able to recover the committed transaction updates. Durability is implemented, in most database systems, by writing transactions to transaction logs (Oliveira da Silva, 2011:7).

3.6.2 Functions, advantages and disadvantages of a DBMS

A DBMS performs essential functions for the data within the database. These functions guarantee integrity and consistency (Morris *et al.*, 2013:23). They are listed below (Connolly & Begg, 2005:12-16; Morris *et al.*, 2013:23-25):

- data dictionary management;
- data storage management;
- data transformation and presentation;
- security management;
- multi-user access control;
- backup and recovery management;
- data integrity management;
- database access languages and application programming;
- database communication interfaces; and
- database utilities.

Although the functions mentioned in the list are common and transparent to the end user, many DBMSs differ from one another in the implementation process (Morris *et al.*, 2013:23). Table 3.10 summarises the advantages and disadvantage of DBMSs.

Table 3.10: Advantages and disadvantages of DBMSs summarised from Connolly and Begg (2005:26-29) and Ramakrishnan and Gehrke (2003:9).

ADVANTAGES	DISADVANTAGES
Data consistency	**Complexity**
Sharing of data	**Cost of DBMS**
Control of data redundancy	**Cost of conversion**
Improved integrity and security	**Performance**
Improved maintenance through data independence	**Higher impact of a failure**
Reduced application development time	

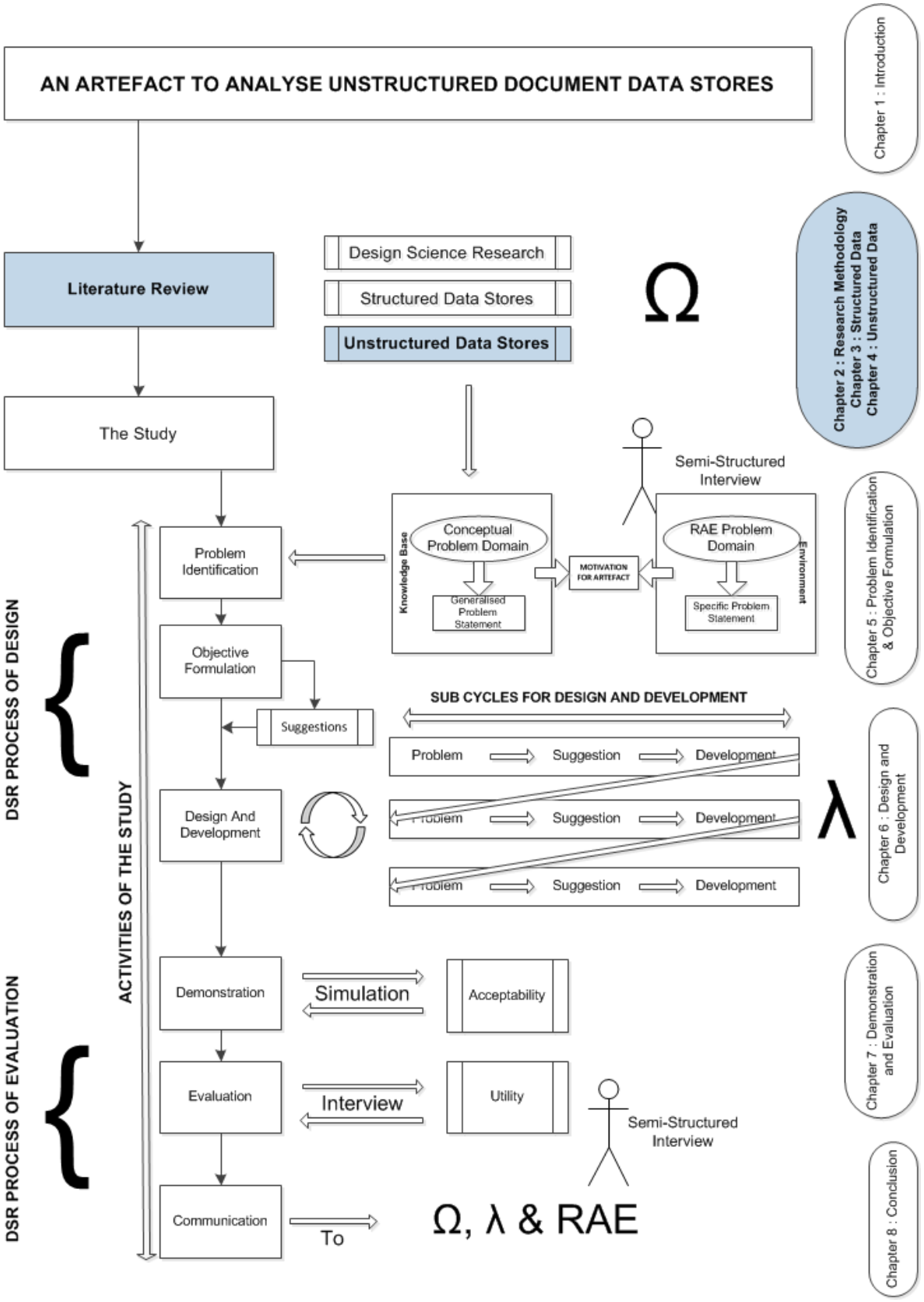
3.7 CHAPTER CONCLUSION

The objective of this chapter was to gain an understanding of RDBMSs. The chapter focused on the relational data model used by these DBMSs. The objective was met by defining databases, discussing the evolution of data models and DBMSs, describing data models, and in-depth discussion of the relational model.

The relational model has been the dominant DBMS since the 1980s and has been widely implemented. The relational model consists of entities, attributes, keys and relationships. Data arranged and fitted into these structures makes the data to be structured simpler and easier to understand. These entities, attributes, keys and relationships form part of the study and help provide a solution to the problem. The characteristics of the relational model are important in the objective formulation and design and development activities of the study.

Throughout the rest of the study, relational DBMSs, traditional DBMSs and structured DBMSs will be referred to as RDBMSs.

The next chapter introduces the concept of NoSQL, the evolution of NoSQL, characteristics implemented by some NoSQL technologies, NoSQL data models and NoSQL technologies.


AN ARTEFACT TO ANALYSE UNSTRUCTURED DOCUMENT DATA STORES
Literature Review
The Study
Design Science Research
Structured Data Stores
Unstructured Data Stores
Ω
Semi-Structured Interview
Problem Identification
Objective Formulation
Suggestions
Design And Development
Demonstration
Evaluation
Communication
Knowledge Base
Conceptual Problem Domain
Generalised Problem Statement
MOTIVATION FOR ARTEFACT
RAE Problem Domain
Specific Problem Statement
Environment
SUB CYCLES FOR DESIGN AND DEVELOPMENT
Problem
Suggestion
Development
λ
Simulation
Acceptability
Interview
Utility
Semi-Structured Interview
To
Ω, λ & RAE
DSR PROCESS OF DESIGN
DSR PROCESS OF EVALUATION
ACTIVITIES OF THE STUDY
Chapter 1 : Introduction
Chapter 2 : Research Methodology
Chapter 3 : Structured Data
Chapter 4 : Unstructured Data
Chapter 5 : Problem Identification & Objective Formulation
Chapter 6 : Design and Development
Chapter 7 : Demonstration and Evaluation
Chapter 8 : Conclusion

CHAPTER FOUR: UNSTRUCTURED DATA STORES

4.1 INTRODUCTION

The primary objective of this study is to develop an artefact which analyses document data stores in terms of structured data model constructs. In order to achieve this, a discussion of the existing literature on unstructured data stores is required because document data stores are unstructured data stores. This literature forms part of the descriptive knowledge of the study.

Tiwari (2011:xvii) states that "The growth of user-driven content has fuelled a rapid increase in the volume and type of data that is generated, manipulated, analysed, and archived". These types of data are derived and gathered from various sources including sensors, Global Positioning Systems (GPS), automated trackers and monitoring systems (Tiwari, 2011:xvii). Businesses, governments, public health organisations and social scientists, to name only a few, have started to pursue data from unconventional sources (Borkar *et al.*, 2012:44). These sources include, amongst others: online buyers' purchases, website interactions, product searches (Borkar *et al.*, 2012:44) web searches, blogs, tweets (Borkar *et al.*, 2012:44; Lohr, 2012) and messages (Lohr, 2012). Therefore, as Borkar *et al.* (2012:44) state, "It is no surprise that support for data-intensive computing, search and information storage is a major challenge in today's computing world". This challenge is a phenomenon labelled 'Big Data'. According to Morris *et al.* (2013:88):

> *"Big Data refers to a movement to find new and better ways to manage large amounts of web-generated data, derive business insight from it, while also providing high performance and scalability at a reasonable cost".*

The term 'Not only SQL' (NoSQL) refers to a next generation database management system (DBMS) that is seen "as non-relational, distributed, open-source and horizontally scalable" (Edlich, 2009). NoSQL DBMSs differ from RDBMSs because

they are not primarily built on table structures and also do not provide SQL for the complex manipulation of data (Moniruzzaman & Hossain, 2013:1).

The first objective of this chapter is to understand NoSQL data models, with specific focus on the document data store model. The second objective is to gain an understanding of NoSQL DBMSs.

This chapter is divided into the following sections: the evolution of Big Data to NoSQL (Section 4.2); NoSQL data models (Section 4.3) distributed NoSQL databases (Section 4.4); NoSQL technologies (Section 4.5); looking beyond towards NewSQL (Section 4.6); and finally, the conclusion (Section 4.7).

4.2 BIG DATA EVOLUTION TO NOSQL

The IT discipline has faced the Big Data challenge for decades. This is because the meaning of 'big' changes continuously. Big in the 1970s meant megabytes (2^{10} kilobytes = 2^{10} x 2^{10} bytes = 2^{20} bytes). Later this became gigabytes (2^{30} bytes) and shortly after terabytes (2^{40} bytes). The current notion of big is petabytes (2^{50} bytes). The University of California estimated that enterprise servers processed 9.57 zettabytes (2^{60} bytes) of data globally in 2008, an amount equal to the daily processing of almost six gigabytes of data for every person in the world (Gopalkrishnan *et al.*, 2012:7). It is believed in the present study that soon the notion of big will become exabytes (2^{70} bytes).

The trend of increasing variety and complexity of data has created an increase in the amount of storage space required to store data. This situation is illustrated in Figure 4.1.

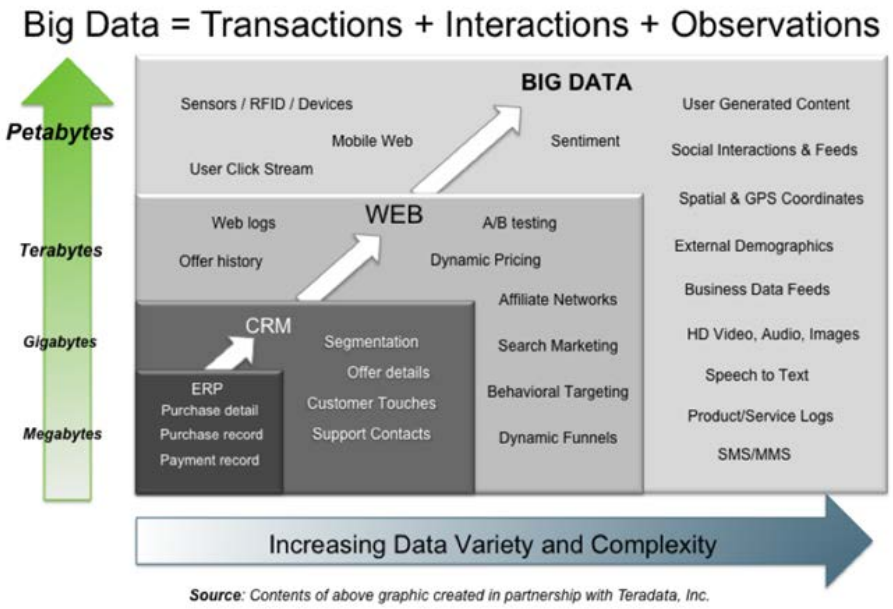


Figure 4.1: Big Data transactions with interactions and observations (Connolly, 2012)

Big Data quickly grew outside the potential of a single computing machine during the relational database revolution. The limited potential of a single computing machine resulted in a new category of parallel DBMSs, known as 'shared-nothing' parallel database systems (PDBSs) (DeWitt & Gray, 1992:85). The adjective 'shared-nothing' (described in Section 4.4.4.1) implies that no memory or storage is shared among the processors (Stonebraker, 1986:4). Parallel DBMSs run on clusters of computers, connected through a network with each computer containing its own resources (Borkar *et al.*, 2012:44). Data are scattered across such a cluster using multiple partitioning strategies. Borkar *et al.* (2012:44) describe how, in these systems, the utilisation of parallel, hash-based divide-and-conquer techniques is implemented to process queries.

In 1980 the first generation of PDBSs appeared with the first commercial solution from Teradata Corporation (Borkar *et al.*, 2012:44). In 2000 a second wave of these systems started to emerge, but these new PDBSs were quickly incorporated by major hardware and software vendors, becoming a successful utilisation of parallel computing systems (Borkar *et al.*, 2012:45). These take-overs implemented a high-level, declarative language, namely SQL, which is commonly found in today's

RDBMSs (Borkar *et al.*, 2012:45). The effect of the implementation of SQL was that it shielded users of PDBSs from the complexities of parallel programming (Borkar *et al.*, 2012:45). With major PDBSs being commercialised, a new set of Big Data problems emerged for the distributed systems world (Borkar *et al.*, 2012:45).

Borkar *et al.* (2012:45) state that “The rapid growth of the World Wide Web and the resulting need to index and query its mushrooming content created big data challenges for search companies such as Inktomi, Yahoo and Google”. Their processing needs are quite different from normal day-to-day transactions and SQL did not offer the solution they required (Borkar *et al.*, 2012:45). PDBS without the implementation of SQL emerged as the hardware platform of choice (Borkar *et al.*, 2012:45).

In the 1980s and 1990s, daily operations were automated using databases in large enterprises (Borkar *et al.*, 2012:45). Due to these automated operations, the database world quickly had to scale up its online transaction processing (OLTP) systems (Borkar *et al.*, 2012:45).

However, at a later stage, as explained by Borkar *et al.* (2012:46), large Web companies in the distributed systems world had to find solutions for achieving very fast and simple lookups as well as updates to large, keyed data sets. These Web companies were driven by user environments that were extremely large (Borkar *et al.*, 2012:46).

RDBMSs that were built for online transaction processing (OLTP) were then disapproved of because they were costly and complex (Borkar *et al.*, 2012:45). Not only did they lack the ability of good horizontal scalability, but they were also not fast enough. Horizontal scalability is known as data and load throughput operations that are shared over multiple servers (Cattell, 2011:12; Pokorny, 2011:278). This is described in Section 4.4.4 under scalability. Through these specific needs, today’s ‘NoSQL movement’ was born (Borkar *et al.*, 2012:45).

In 1998, the term NoSQL was first introduced and used to describe Carlo Strozzi's open-source relational database (Lith & Mattsson, 2010:15). His database did not provide a SQL interface (Lith & Mattsson, 2010:15). The meaning of the term has changed since 1998 and is widely used to describe these new NoSQL DBMSs.

The first reference to the term NoSQL after 1998 was in early 2009 in a blog written by Eric Evans, when he announced that an event had been organised to discuss open-source distributed databases (Evans, 2009). As a result of the blog, the name NoSQL was used to label these new DBMSs, which are non-relational and distributed data stores. These databases did not allow atomicity, consistency, isolation and durability (ACID), which are the key properties of RDBMSs. It should be mentioned that these systems were developed to provide horizontal scalability (Cattell, 2011:12). Horizontal scalability is discussed in Section 4.4.4.

The main motivation for the implementation of NoSQL was the applications from the Web 2.0 (described in Section 4.2.1) domain (Hecht & Jablonski, 2011:336; Okman *et al.*, 2011:541; Moniruzzaman & Hossain, 2013:1). A second motivation that led to NoSQL was the ever-changing nature of the data world (Helland, 2011:47). Multiple tables had to be used for storing the different types of web content such as text, comments, pictures, videos and other kinds of information (Hecht & Jablonski, 2011:336). The capabilities and possibilities of RDBMS were not suited for these applications and the RDBMS schema became a burden for the web applications (Hecht & Jablonski, 2011:336). RDBMS's full ACID support (discussed in Chapter 3) joins (discussed in Chapter 3) and locks (discussed in Section 4.4.6) influenced performance that was not suited for the Web 2.0 domain (Cattell, 2011:12). As an example, system unavailability became prominent in RDBMSs when a simple feature needed to be added or removed from a blog (Hecht & Jablonski, 2011:336).

Two major problems led to the consideration of NoSQL databases as listed by Moniruzzaman and Hossain (2013:2):

1. Data not only generated by users, systems and sensors, but also the acceleration of big distributions systems, such as cloud services, presented an exponential growth in the volume of data.
2. The internet, Web 2.0, social networks and access to the data of various systems generated an accelerated increase in the interdependency and complexity of data.

A variety of NoSQL systems have emerged in recent years and have been developed to provide horizontal scalability for the simple transactions of reading from and writing to the database that is distributed over several servers (Cattell, 2011:12). NoSQL DBMSs generally support the following features (Cattell, 2011:12):

- "the ability to horizontally scale 'simple operation' throughput over many servers;
- the ability to replicate and to distribute (partition) data over many servers;
- a simple call level interface or protocol (in contrast to a SQL binding);
- a weaker concurrency model than the ACID transactions of most relational (SQL) database systems;
- efficient use of distributed indexes and RAM for data storage; and
- the ability to dynamically add new attributes to data records".

The ability to dynamically add attributes and data to records (Cattell, 2011:12) becomes critical when these records are unstructured. For example, one record may contain ten attributes while others may contain five or fewer. If the records are unstructured, the data become unstructured. Unstructured data are stored in different data models, which are specifically based on the data, and at times, the relationships needed.

Figure 4.2 is a representation of unstructured data. In the figure it can be seen that there are various record formats containing different attributes. For example, record one contains three attributes, while record two contains five attributes. It should also be noted that the attributes in a specific column are not the same. For example, COLUMN 1 contains a customer identifier, surnames, names, ID numbers and street

addresses, while COLUMN 3 contains surnames, e-mail addresses and phone numbers. This clearly indicates that the records are different from one another and do not conform to a fixed structure.

COLUMN 1	COLUMN 2	COLUMN 3	COLUMN 4	COLUMN 5
CUS101101	Kelly	Mac Pearson		
Mac Pearson	Justin	jtt.rud@gmail.com	8412155102057	0111245778
Alice	Creswell	0112124545		
7501021002085	Alex	Seamen	0211216532	
Inmon	Kate	kate. inmon @gmail.com	kate. inmon @muv.ac.za	0111245778
2509015324854	Alex			
Alexton Road	Vanderbijlpark	De Jager		
Ring Road	Vereenining	Smith	M	

Figure 4.2: Unstructured data example with multiple attributes

The example in Figure 4.2 is relatively simple since only a few records are displayed which could easily be viewed and interpreted by humans. However, the same cannot be said when there are millions or billions of records with hundreds of attributes each. Reviewing extracted data in the real application environment in Appendix A illustrates the view and interpretation complexity.

NoSQL data stores support various data models that are unique and different from RDBMSs (Leavitt, 2010; Cattell, 2011; Moniruzzaman & Hossain, 2013). The main difference between the two DBMS technologies is that NoSQL stores mostly unstructured data, which are more complicated to interpret. The data stores of RDBMSs are relational and store structured data. Typical data models used in NoSQL DBMSs include: key-value stores, document stores, column-oriented stores and graph databases (Cattell, 2011:12; Hecht & Jablonski, 2011:337; Moniruzzaman & Hossain, 2013:4). These data models are further explored in Section 4.3 after the discussion on Web 2.0 that follows.

4.2.1 Web 2.0

Web 2.0 refers to the dynamic way of creating, developing and using applications over the internet compared to the traditional static pages of Web 1.0 (Cervinschi & Butucea, 2010:39).

The Web was the grand vision of Tim Berners-Lee, who in 1989 started to develop the World Wide Web (WWW) (Richardson, 2009:1). The original vision of Tim Berners-Lee's World Wide Web was the creation of a collaborative medium, where all could meet, read and write (Carvin, 2005).

In 1999, Web 2.0 was first mentioned by DiNucci (1999). DiNucci (1999) states that Web 2.0 is "The Web we know now, which loads into a browser window in essentially static screenfulls, is only an embryo of the Web to come. The first glimmerings of Web 2.0 are beginning to appear, and we are just starting to see how that embryo might develop".

At a later stage in a blog by John Robb (2003)[4], quoted by Cervinschi and Butucea (2010:39), Web 2.0 was described as "A system that breaks with the old model of centralized Web sites and moves the power of the Web/Internet to the desktop".

The term was comprehensively defined by O'Reilly (2005) as:

> *"Web 2.0 is the network as platform, spanning all connected devices; Web 2.0 applications are those that make the most of the intrinsic advantages of that platform: delivering software as a continually-updated service that gets better the more people use it, consuming and remixing data from multiple sources, including individual users, while providing their own data and services in a*

[4] The source John Robb (2003) is no longer available via the URL provided by Cervinschi and Butucea (2010). Therefore the complete reference to the source is excluded from the reference list.

form that allows remixing by others, creating network effects through an 'architecture of participation', and going beyond the page metaphor of Web 1.0 to deliver rich user experiences".

During 2006, after numerous comments on his 2005 definition, O'Reilly (2006) reformulated the definition of Web 2.0 as:

"Web 2.0 is the business revolution in the computer industry caused by the move to the internet as platform, and an attempt to understand the rules for success on that new platform. Chief among those rules is this: Build applications that harness network effects to get better the more people use them".

Both the definitions of O'Reilly (2005; 2006) are very abstract and therefore this study provides a simpler notion of Web 2.0 as described by Cervinschi and Butucea (2010:39),Tim Berners-Lee (Carvin, 2005) and the notion of Big Data. In this study term Web 2.0 refers to the creation and development of applications that are used over the internet for the generation, integration, and collaboration of dynamic data.

The primary objective of this chapter is to gain an understanding of NoSQL data models, and the next section explores these models.

4.3 NOSQL DATA MODELS

As opposed to the single relational data model of the RDBMS, NoSQL DBMSs make provision for multiple data models. Most of these data models are unstructured, storing attribute-value pairs using different data model structures. The most common NoSQL data models are: key-value stores, document stores, column-oriented stores and graph databases (Cattell, 2011:12; Hecht & Jablonski, 2011:337; Moniruzzaman & Hossain, 2013:4).

The running examples used in this chapter are related to the examples used in Chapter 3. Figures 4.3 and 4.4 are the related examples from Chapter 3. The two data tables in Figure 4.3 represent the data used in Chapter 4 to demonstrate the various NoSQL data model representations where possible. The first table “CUSTOMER” is related to the second table “INVOICE” by the common attribute “CUSTOMER_ID”. This enables the user to see what purchases have been made by each customer by matching the records.

Table name: CUSTOMER

CUSTOMER_ID	NAME	SURNAME	ID_NUMBER	PHONE
CUS101101	Kelly	Mac Pearson	8802021243056	0169521122
CUS101201	Justin	Mac Pearson	8412155102057	0111245778
CUS106102	Alice	Creswell	5508250012075	0112124545
CUS203101	Alex	Seamen	7501021002085	0211216532

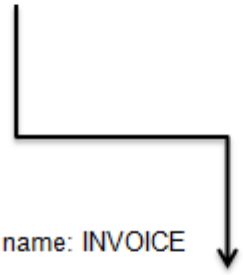


Table name: INVOICE

INVOICE_NUM	CUSTOMER_ID	DATE	TOTAL
INV0012	CUS101101	2012-01-02	417-00
INV0013	CUS101201	2012-02-29	25-00
INV0014	CUS203101	2013-05-25	142-00
INV0032	CUS101201	2013-08-02	925-00
INV0035	CUS101201	2013-10-16	2923-00
INV0065	CUS101101	2013-11-16	545-00
INV0066	CUS106102	2013-11-17	1925-00

Figure 4.3: Example data representation of the relational model

Figure 4.4 is an extended data example of Figure 4.3 used for this chapter. In this example it can be seen that one “INVOICE” contains many “INVOICE_LINE” records; this forms a one-to-many relationship between “INVOICE” and “INVOICE_LINE”. From the example it can also be seen that many “PRODUCT” records are contained in many “INVOICE_LINE” records; this forms a one-to-many relationship between “PRODUCT” and “INVOICE_LINE”.

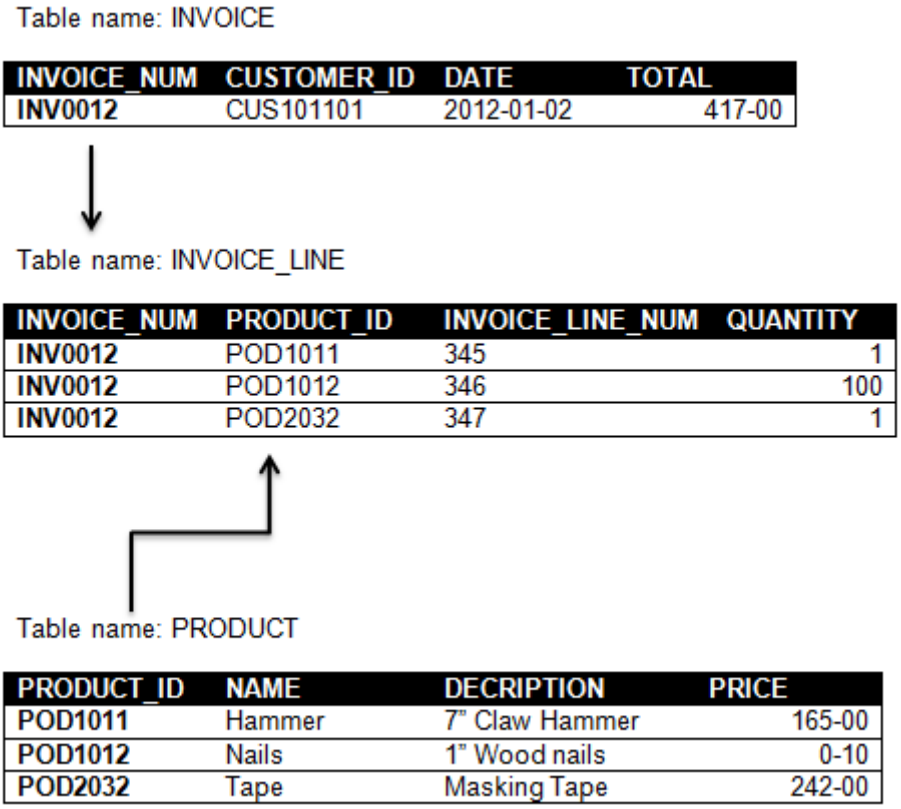


Figure 4.4: Example data representation of one-to-many relationship between "INVOICE", "INVOICE_LINE" and "PRODUCT"

JavaScript Object Notation (JSON) is an important data representation structure implemented by some of the NoSQL data models. Therefore a discussion of JSON follows to provide clarity for JSON implementations.

4.3.1 JavaScript Object Notation (JSON)

JavaScript Object Notation, or JSON (pronounced "jay-sun"), is a syntax that is used for storing and exchanging text information (Crockford, 2006; Crockford, 2009). JSON is similar to Extensible Mark-up Language (XML) in that both provide a standard for syntax that both humans and computer programs are able to understand. XML focuses on document layout while JSON focuses on data representation. The storage and retrieval of data values in their original form is simplified with JSON. It removes the need for any type of mapper or specific code

converter (Membrey *et al.*, 2010:11). The syntax is independent of a programming language. JSON is built on two structures (Crockford, 2006; Crockford, 2009):

1. Collection of name/value pairs – seen as an object, record, instance, structure, keyed list or associative array as demonstrated in Figure 4.5.
2. Ordered list of values – seen as an array, vector, list or sequence as demonstrated in Figure 4.6.

Most programming languages support these common data structures in one form or another. JSON is used to describe unstructured data in this study.

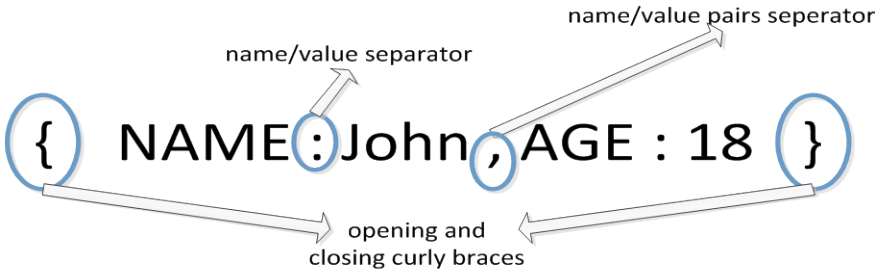


Figure 4.5: Example of an object in JSON

Figure 4.5 is an example of an object in JSON. In Figure 4.5, the opening curly bracket acts as the start of the object and the closing curly bracket as the end of the object. The colon acts as the name/value separator and the comma acts as the name/value pair separator.

Figure 4.6 is an example of an array in JSON. In Figure 4.6, the opening square bracket acts as the start of the array and the closing square bracket as the end of the list. The comma is the value separator between the values in the array.

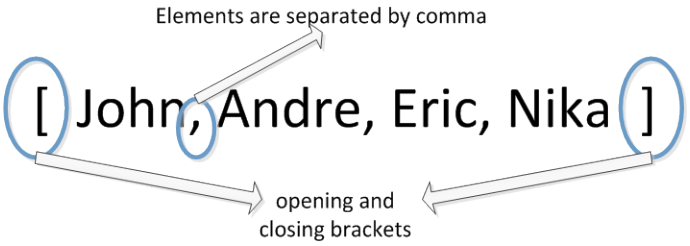


Figure 4.6: Example of an array in JSON

Figure 4.7 is an example of where an object is incorporated in the array in JSON. In Figure 4.7, the array “people” contains two objects with attributes “age” and “name”.

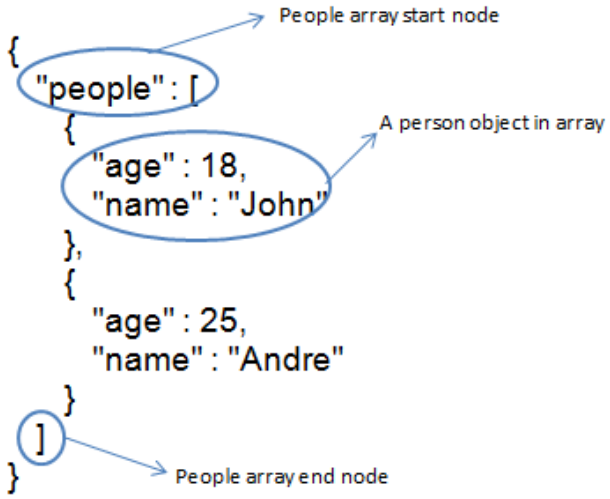


Figure 4.7: Example of an array incorporating objects in JSON

The next sections explore the four NoSQL data models.

4.3.2 Key-value stores

A key-value store is a collection of key and data value pairs. The key acts as a pointer to a record with a value (Cattell, 2011:14). A key value store can accommodate any data structure possible for that specified key (Hecht & Jablonski, 2011:337). This works similar to maps or dictionaries where unique keys are used to identify the data values (Hecht & Jablonski, 2011:337). The key acts as the only pointer to retrieve the data since the data are not interpretable as such to the system (Hecht & Jablonski, 2011:337).

Key-value stores are the most widely used models out of the four mentioned and present the opportunity for a program to store data in a schema-less or unstructured manner. Key-value stores have existed for a long time, for example, in 1991 Berkeley D.B. from Oracle developed a key-value store (Olson *et al.*, 1999:1). However, it only became widely used in the last few years due to the influence of Amazon's Dynamo database published in 2007.

Hecht and Jablonski (2011:337) note that there is a possibility of adding some structure to key-value stores by grouping the key value-pairs into collections. Usually these key-value stores, which consist of key-value pairs, are implemented by distributed hash tables (Stonebraker, 2010b:10).

Figure 4.8 is a graphical representation of a key-value data example. In Figure 4.8 the "KEYS" are searchable and represent the index used to point to the record. The "VALUES" are not searchable by the system.

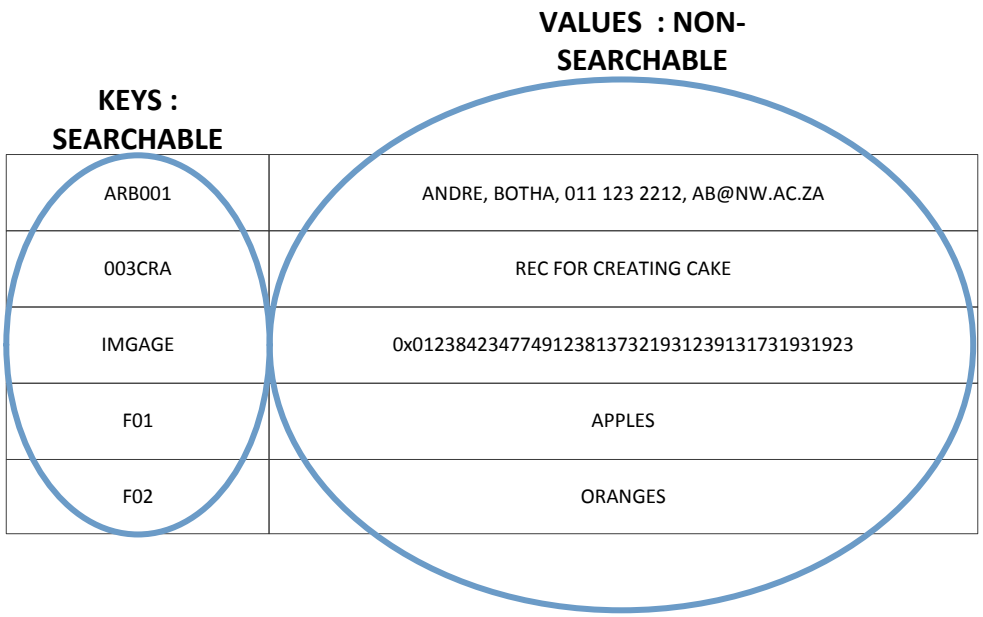


Figure 4.8: Key-value store graphical representation

Figure 4.9 is a key-value representation of some of the data in Figure 4.3. Various other representations are possible for these available data as the data are unstructured. In Figure 4.9, it can be seen that there are numerous entries of “CUSTOMER” and “INVOICE”. It can also be seen that the values for these entries differ from one another: some are primitive data (CUS101101, INV0012 and INV0013), while others have been formatted to include attribute names with the data (CUS106102 and INV0066).

KEY	VALUE
CUS101101	Kelly,Mac Pearson,8802021243056,0169521122
INV0012	CUS101101;2012-01-02;1865-00
INV0013	CUS101201;2012-02-29;25-00
CUS106102	NAME:Alice,SURNAME:Creswell,ID_NUMBER:5508250012075,PHONE0112124545
INV0066	CUSTOMER_ID:CUS106102, DATE:2013-11-17,TOTAL 1925-00

Figure 4.9: Key-value store representation of Figure 4.2

Key-value stores can maintain almost any structure possible, therefore JSON structures are also possible in key-values stores, as in the case of Riak (Basho,

2013). Other examples of formats that Riak may include is XML, Hypertext Markup Language (HTML), user data, text, images and log files (Basho, 2013).

Key-value data stores are stored mostly in memory, and therefore they are used for caching of time-intensive SQL queries (Hecht & Jablonski, 2011:337). Moniruzzaman and Hossain (2013:5) maintain that key-value stores are “ideally suited to lightning-fast, highly-scalable retrieval of the values needed for application tasks like managing user profiles or sessions or retrieving product names”.

Key-value stores favour high scalability (discussed in Section 4.4.4) when compared to consistency (discussed in Sections 4.4.1 and 4.4.2). In order to favour this high scalability, rich ad hoc queries and analytical features are omitted (Strauch *et al.*, 2011:52). Keys are sometimes limited in length and data type, but there are fewer limitations on the data values in terms of length and data type (North, 2009). The data values of key-value stores are isolated and independent, and therefore relationships between data values of different keys may only be managed in programming logic (Hecht & Jablonski, 2011:337). Insert, delete and lookup operations are supported in all key-value stores (Cattell, 2011:16).

4.3.3 Document stores

Document stores are similar to key-value stores, with the exception that they can handle multi-attribute lookups that may have completely different forms of key-value pairs (Hecht & Jablonski, 2011:337). Document stores may further contain nested documents and lists (Cattell, 2011:14). The documents are indexed, and a simple query method is provided to query the data (Cattell, 2011:14). Each document carries an exceptional key identifier that has to be unique and identifies the document (Cattell, 2011:16).

The inspiration for document stores came from Lotus Notes (Moniruzzaman & Hossain, 2013:5). Lotus Notes was originally designed to store documents as the name implies (Moniruzzaman & Hossain, 2013:5). However, confusion is caused

when the term “document store” is used, because it can actually keep any pointer-less object and not only store documents (Moniruzzaman & Hossain, 2013:5) such as articles or MSWord files in the traditional sense as described by Cattell (2011:16).

Document stores are similar to key-value stores, as neither have any schema or structure restrictions, so they may contain and can easily add any attribute and additional attributes to the existing document during runtime (Hecht & Jablonski, 2011:337). Collections (the grouping of similar documents) and indexes that are defined in these collections are the only structures that are predefined (Cattell, 2011:18). Some of the document stores in DBMSs provide the ability to support secondary indexes and various nested document types or lists per database (Cattell, 2011:16).

Document stores differ from key-value stores in the sense that they encapsulate key-value pairs in formats such as Extensible Mark-up Language (XML) (Moniruzzaman & Hossain, 2013:5), JavaScript Object Notation (JSON) and JSON-like documents (Hecht & Jablonski, 2011:337; Moniruzzaman & Hossain, 2013:5). The data within documents are searchable and may be queried, unlike the values in key-value stores (Moniruzzaman & Hossain, 2013:5).

Figure 4.10 is a simple representation of a document data store example. In Figure 4.10, the “Document ID” is the identifier for the document and the JSON documents are data for that “Document ID”. Attributes such as mail, phone, etc. are searchable. The “Document ID” is generated by the NoSQL DBMS.

Figures 4.11 and 4.13 are different representations of the data in Figures 4.3 and 4.4 respectively using JSON for document stores. Figures 4.12 and 4.14 are table representations of Figures 4.11 and 4.13.

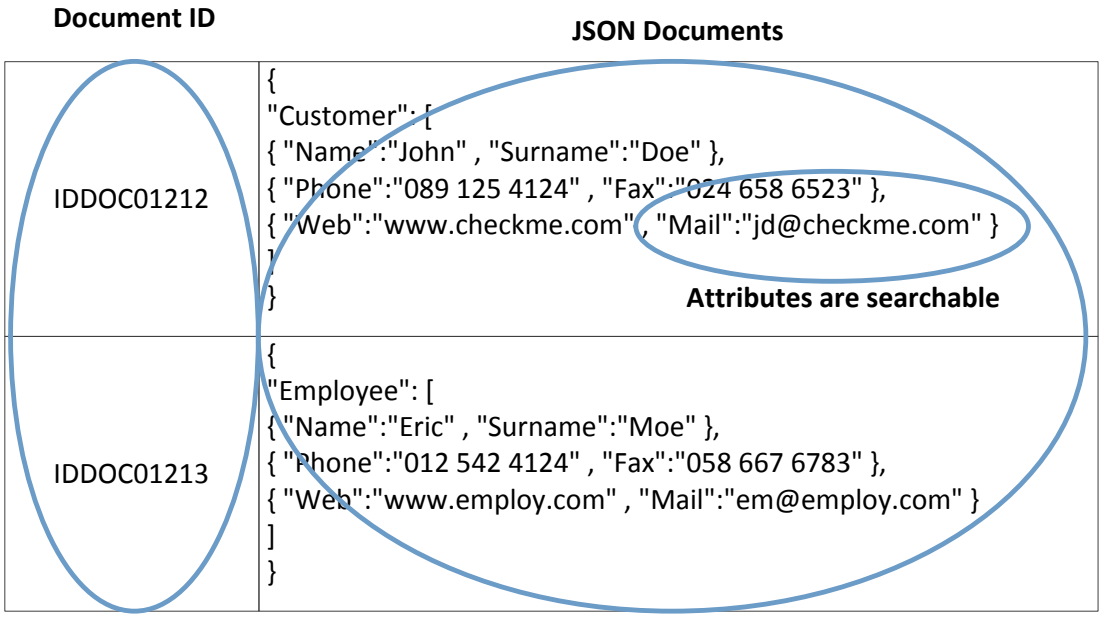


Figure 4.10: Document store graphical representation using JSON

Figure 4.11 demonstrates a customer and all invoices for that specific customer. “CUSTOMER” contains attributes such as “ID_NUMBER”, “NAME”, “SURNAME”, etc., and most importantly a list object (“INVOICES”). These “INVOICES” form a list of “INVOICE” objects (sub-documents) of the “CUSTOMER” object (document). They contain attributes such as “INVOICE_NUM”, “DATE” and “TOTAL”. This specific data record example can be described as a one-to-many relationship between “CUSTOMER” and “INVOICES”.


```
{
  "CUSTOMER_ID" : "CUS101101",
  "ID_NUMBER" : "8802021243056",
  "INVOICES" : [
    {
      "DATE" : "2012-01-02",
      "INVOICE_NUM" : "INV0012",
      "TOTAL" : "417-00"
    },
    {
      "DATE" : "2013-11-16",
      "INVOICE_NUM" : "INV0065",
      "TOTAL" : "545-00"
    }
  ],
  "NAME" : "Kelly",
  "PHONE" : "0169521122",
  "SURNAME" : "Mac Pearson"
}
```


Figure 4.11: First raw JSON representation of Figure 4.3

Figure 4.12 is a table-like representation of Figure 4.11. In Figure 4.12, the invoice column represents associated “INVOICES” for that specific “CUSTOMER”. It should be noted that although the tables for the different records in this example are of the same format, it need not be the case. In other words the invoice detail per customer need not have the same format for all the customers. The different formatting of the lines of the tables in Figure 4.12 emphasises this characteristic of document data stores.

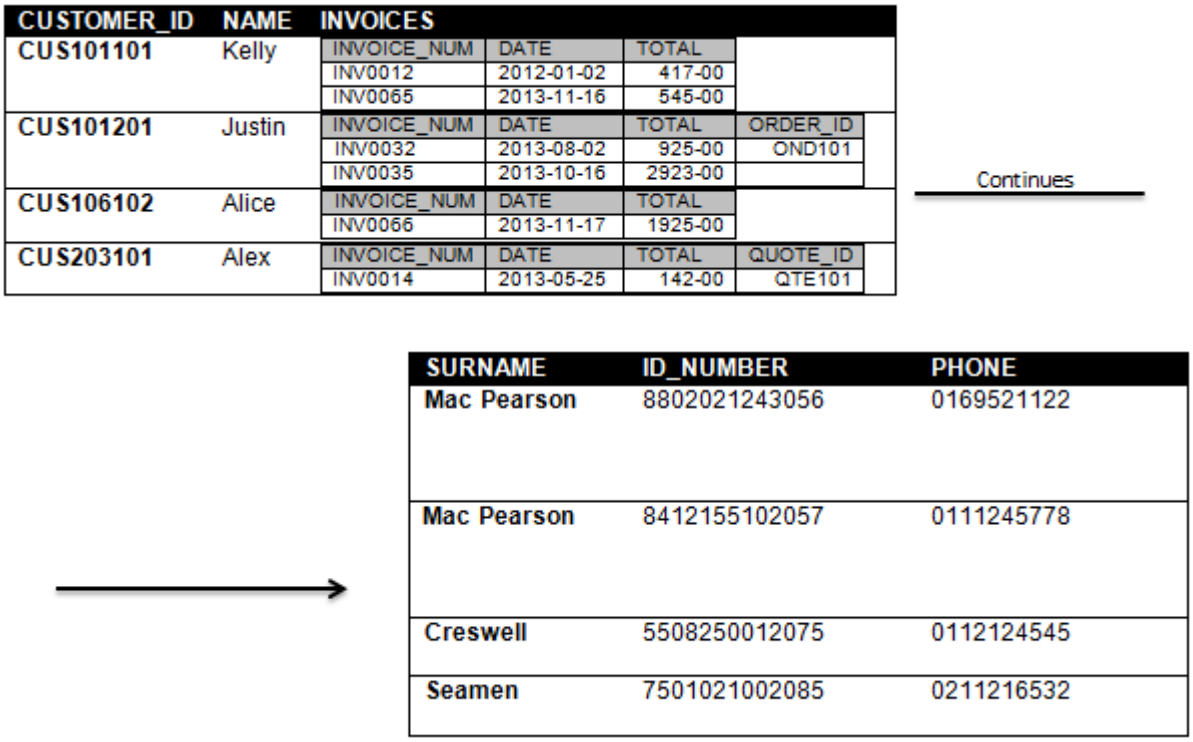


Figure 4.12: Table representation of Figure 4.11

Figure 4.13 demonstrates an invoice containing the specific customer information and all items that were purchased taken from the data of Figure 4.4. The invoice contains attributes such as “INVOICE_NUM”, “DATE”, “TOTAL”, a sub-document “CUSTOMER” and an object list “ITEMS”. “CUSTOMER” is a single sub-document containing attributes such as “ID_NUMBER”, “NAME”, “SURNAME”, etc. “ITEMS” is an object list containing objects of “ITEMS” and contains attributes such as “INVOICE_LINE_NUM”, “PRODUCT_ID” and “QUANTITY”. This specific data record

example can be described as a one-to-one relationship between “INVOICE” and “CUSTOMER”, but a one-to-many relationship between “INVOICE” and “ITEMS”.


```
{
  "INVOICES" : [
    {
      "CUSTOMER" : {
        "CUSTOMER_ID" : "CUS101101",
        "ID_NUMBER" : "8802021243056",
        "NAME" : "Kelly",
        "PHONE" : "0169521122",
        "SURNAME" : "Mac Pearson"
      },
      "DATE" : "2012-01-02",
      "INVOICE_NUM" : "INV0012",
      "ITEMS" : [
        {
          "INVOICE_LINE_NUM" : "345",
          "PRODUCT_ID" : "POD1011",
          "QUANTITY" : "1"
        },
        {
          "INVOICE_LINE_NUM" : "346",
          "PRODUCT_ID" : "POD1012",
          "QUANTITY" : "100"
        },
        {
          "INVOICE_LINE_NUM" : "347",
          "PRODUCT_ID" : "POD2032",
          "QUANTITY" : "1"
        }
      ],
      "TOTAL" : "417-00"
    }
  ]
}
```


Figure 4.13: Second raw JSON representation of Figures 4.3 and 4.4

Figure 4.14 is a table-like representation of Figure 4.13. In Figure 4.14, “CUSTOMER” is a sub-document of “INVOICE”. The “ITEMS” column forms an object list of sub-documents for the items purchased for a specific “INVOICE”.

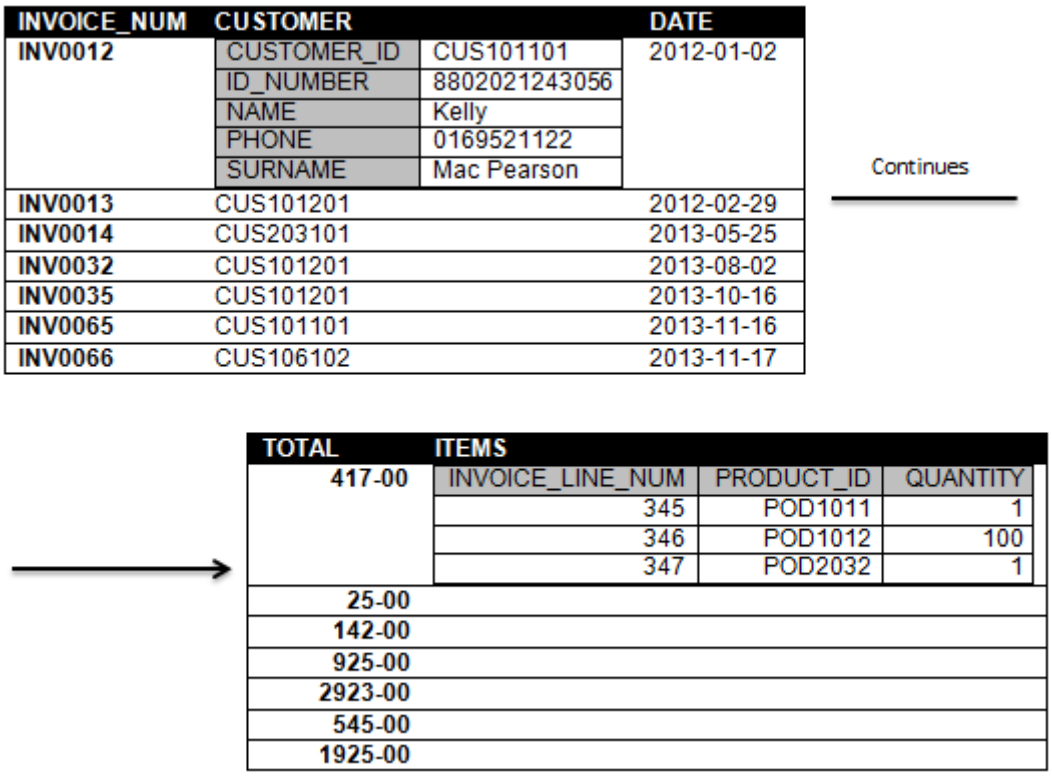


Figure 4.14: Nested table representation of Figure 4.12

As can be seen from these figures and examples, there are quite a number of possibilities for representing the data. In the first example, Figure 4.11 and Figure 4.12 represent the “CUSTOMER” and “INVOICE” data. In the second example, Figure 4.13 and Figure 4.14 represent the same data in a different format but expand these data further by incorporating “ITEMS” into the structure.

Hecht and Jablonski (2011:337) state that document stores are convenient for “data integration and schema migration tasks”. Maintaining document stores is easy and more desirable for flexible web applications (Hecht & Jablonski, 2011:337). Real-time analytics, logging and the storage layer of small and flexible websites like blogs are popular applications of document stores. Moniruzzaman and Hossain (2013:5) state that document stores “are good for storing and managing Big Data-size collections of literal documents, like text documents, email messages, and XML

documents, as well as conceptual documents like de-normalized (aggregate) representations of a database entity such as a product or customer".

The most prominent document stores listed are CouchDB and MongoDB (Hecht & Jablonski, 2011:337; Moniruzzaman & Hossain, 2013:5).

Hecht and Jablonski (2011:337) and Basho (2013) name Riak as a document store; however, this is not true as Cattell (2011:15) points out. Riak does not include some of the important features found in documents stores and is defined by Cattell (2011:15) as an "advanced key-value store" and by Moniruzzaman and Hossain (2013:5) as a key-value store. This study also defines Riak as key-value store DBMSs.

Weaker concurrency and atomic properties are found in document stores and they do not provide explicit locks (Cattell, 2011:18). Concurrency control is discussed in Section 4.4.5.

4.3.4 Column-oriented stores

Column-oriented stores are also similar to key-value stores, with the exception that an attribute of a dataset is stored in a column-oriented manner and not in a row orientation (Matei, 2010:5). Column-oriented stores are also referred to as Extensible Record Stores (Cattell, 2011:14; Hecht & Jablonski, 2011:337), Wide-Column stores and Column-Family stores (Hecht & Jablonski, 2011:337; Moniruzzaman & Hossain, 2013:4). Column-oriented stores can store multiple attributes per identifier key (Moniruzzaman & Hossain, 2013:6). In column-oriented stores, an arbitrary number of key value pairs can be stored in the map (Hecht & Jablonski, 2011:337).

Data values that change are not overwritten. In order to achieve versioning, better performance and consistency, a chronological order is used to store multiple versions of a value (Hecht & Jablonski, 2011:337). It is especially beneficial for data

organisation and its partitioning when columns can be grouped together into column families (Hecht & Jablonski, 2011:337). Users are sometimes able to group columns together in terms of semantics (logical meaning of data). Such column groups are pre-defined and known as column families (Cattell, 2011:20). These grouping into column families forms units of access control (Chang *et al.*, 2008:3). Although column-oriented stores are less flexible than key-value stores and document stores (since column families have to be predefined) rows and columns may still be added at runtime (Hecht & Jablonski, 2011:337).

Google's Bigtable was the inspiration for all these stores. Hecht and Jablonski (2011:337) state that "Bigtable is used in many Google projects varying in requirements of high throughput and latency-sensitive data serving". Chang *et al.* (2008:4) define Bigtable as a "distributed storage system for managing structured data that is designed to scale to a very large size". Building on this is the description that Bigtable is a sparse, distributed, persistent multidimensional sorted map (Chang *et al.*, 2008:4). Moniruzzaman and Hossain (2013:5) maintain that column-orientation is good for "distributed data storage", "large-scale, batch-oriented data processing" and "exploratory and predictive analytics performed by expert statisticians and programmers".

The graphical representation of column-oriented stores is similar to that of relational databases: a table structure. The main difference between the two lies in how they handle null values. In relational databases, a null value would be stored in a column when there are no data available for that attribute. In the case of column-oriented stores, a null value will be stored within a key value pair if that dataset needs it for other purposes, otherwise it will be omitted when the attribute is not applicable to the specific record (Hecht & Jablonski, 2011:337).

Figure 4.15 is a graphical representation of a column-oriented store data example.

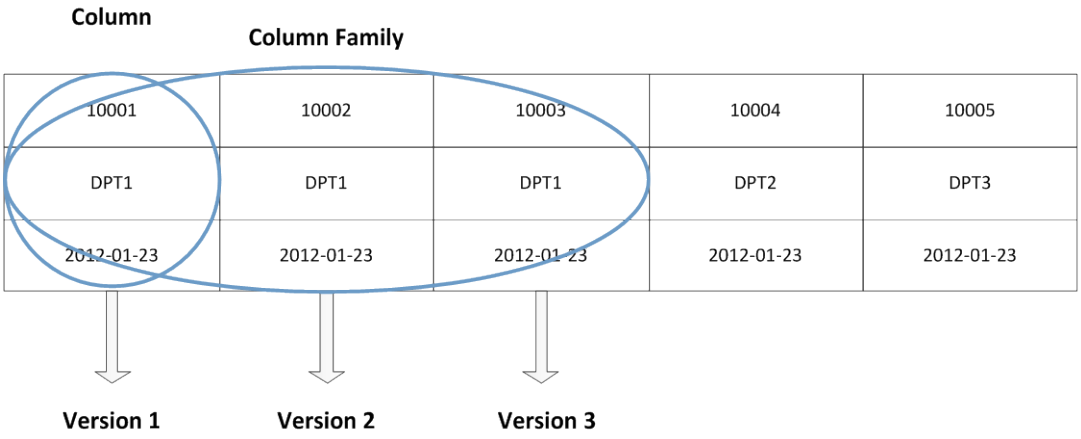


Figure 4.15: Column-oriented graphical representation

Figure 4.16 is a column-oriented representation of Figure 4.3. In Figure 4.16, “10000001”, “10000002”, “10000003” identify the different versions of the data, while “KMP” and “CUSTO” identify in which column family the columns are grouped.

10000001	10000002	10000003	10000004	10000005
CUSTO	CUSTO	CUSTO	CUSTO	CUSTO
KMP	KMP	KMP	JMP	JMP
2013-3-3	2013-4-3	2013-6-3	2013-1-3	2013-2-3
Kelly	Kelly	Kelly	Justin	Justin
Mac Pearson	Mac Pearson	Mac Pearson	Mac Pearson	Mac Pearson
8802021243056	8802021243056	8802021243056	8412155102057	8412155102057
0169521122	0169521122	0169521122	0111245778	0111245778

Figure 4.16: Column-oriented representation of Figure 4.2

Open-source implementations such as HBase and Hypertable are similar to Bigtable. The Cassandra implementation differs from HBase and Hypertable, because Cassandra implements supercolumns (Hecht & Jablonski, 2011:337). Multiple columns are contained in supercolumns and may be stored as column families. This makes the Cassandra implementation more desirable for handling complex and expressive data structures (Hecht & Jablonski, 2011:337). The aggregation capabilities of column-oriented stores are used well in Oracle Essbase, an Online

Analytical Processing (OLAP) implementation, or in data warehouses (Han *et al.*, 2011:364; Weber, 2011?:5)[5].

Column-oriented stores are similar to key-value stores where extra features also have to be managed with programming logic (Hecht & Jablonski, 2011:337). Since the values cannot be translated within the system, relationships are not supported among the datasets and strings are the only data type supported (Hecht & Jablonski, 2011:337).

4.3.5 Graph databases

The last of the NoSQL DBMSs data models is graph databases. These databases are focused on the effective management of linked data (Hecht & Jablonski, 2011:337). Graph databases are built on the graph theory (Weber, 2011?:6; Robinson *et al.*, 2013:1), where edges connect vertices. Similar to the way in which roads connect cities, relationships are the edges in databases that connect entities. Robinson *et al.* (2013:2) explain that *"*graphs represent entities as nodes and the ways in which those entities relate to the world as relationships*"* and

> *"a graph database management system (henceforth, a graph database) is an online database management system with Create, Read, Update and Delete methods that expose a graph data model".*

Applications with heavily linked data with many relationships are suitable for graph databases according to Hecht and Jablonski (2011:338). This is because traversals efficiently replace cost-intensive operations such as recursive joins (Hecht and Jablonski (2011:338).

[5] The source of Weber (2011?) is not formally linked to a specific date and therefore the publication date estimated by this study is 2011 based on content.

Graph databases are generally schema free (Robinson *et al.*, 2013:96). These are the only data models that are concerned with relations and the visual representation of information (Moniruzzaman & Hossain, 2013:7).

Direct adjacency lists are most commonly used for storing context of edges and vertices (Robinson *et al.*, 2013:2). In these adjacency lists, the vertices describe which other vertices they are linked with (Weber, 2011?:6).

Figure 4.17 contains example data used to demonstrate a graph in Figure 4.18. In Figure 4.17, the table is a matrix representation where the objects are named in the top row and first column and the relations between these objects are identified in the rest of the table. Figure 4.18 is a graphical representation of a graph database data example. In Figure 4.18, the vertices are entities that are connected with edges representing relationships. For example, Martin, a person entity, is friends with Eric, another person entity. Eric, a person entity, works at University B, which is a building or location entity. In this manner, highly related data are built.

OBJECTS	Martin	Chanelle	Eric	Susan	Rugby	Tennis	University A	University B
Martin		Friends			Playes			
Chanelle	Friends		Married	Friends				
Eric	Friends	Married	Friends		Watches	Coach	Studies At	Works At
Susan		Friends						Studies At
Rugby								
Tennis								
University A						Presents		
University B								

A = {(rugby, martin), (rugby,eric), (martin, chanelle), (martin, eric),(eric, tennis), (tennis, university a),(eric, university a), (channel, susan), (eric, university b), (susan, university b) }

Figure 4.17: Graph database data example

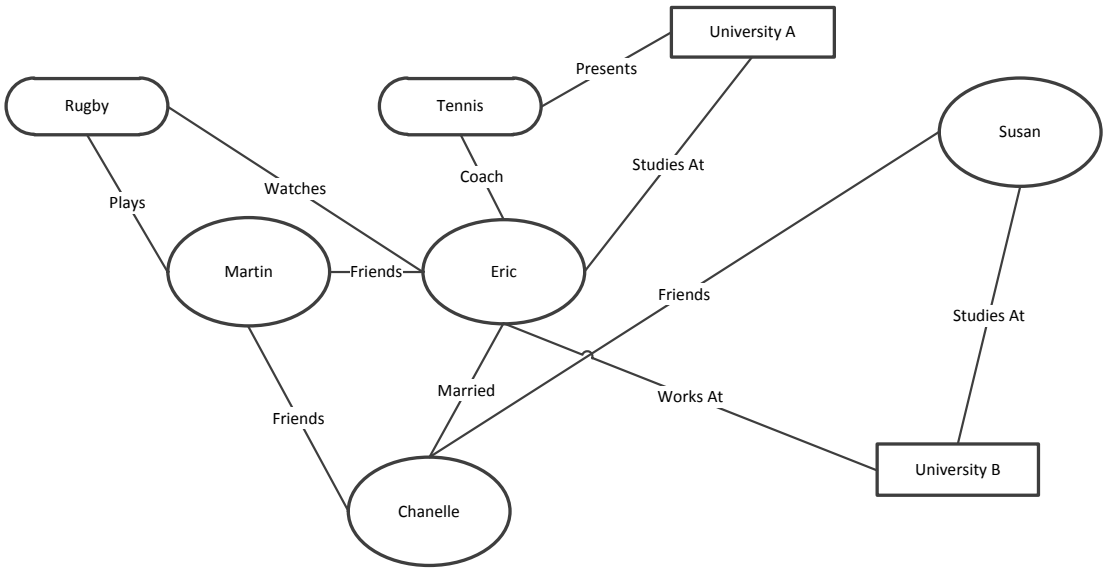


Figure 4.18: Graph database data representation example

Graph databases are well suited for location-based services, finding common friends on social networks, finding the shortest path between two subjects, and knowledge representations (Weber, 2011?:6). Moniruzzaman and Hossain (2013:5) state that graph databases are good for "when you are more interested in relationships between data than in the data itself: for example, in representing and traversing social networks, generating recommendations (e.g., upsell or cross-sell suggestions), or conducting forensic investigations (e.g. pattern detection)".

A common feature found in key-value stores, document data stores and column-oriented data stores is that in order to enhance distribution, they store denormalised data. Many enquiries, execution penalties and complex code in the program layer will arise if databases like these should be used to address relational data requirements (Hecht & Jablonski, 2011:337)

Cattell (2011), Han *et al.* (2011) and Hecht and Jablonski (2011) state that the key characteristics of NoSQL DBMSs are that they store unstructured or schema-less data.

4.4 DISTRIBUTED NOSQL DATABASES

NoSQL databases are described as distributed and horizontally scalable, therefore it is essential to investigate these architectures.

This section discusses distributed NoSQL DBMSs, an integral part of these technologies. Distributed database concepts discussed here include the:

- ACID properties for concurrency on RDBMSs summary;
- CAP theorem that guides NoSQL database concurrency management;
- properties of BASE which is the relaxation ACID found in RDBMSs;
- scalability, the core architecture that is provided by these databases;
- programming strategy known as MapReduce that distributes data across multiple nodes; and
- mechanisms of concurrency control to assist in the concurrent access to data.

These characteristics do not constitute an essential part of this study, but it is necessary to create context to facilitate the understanding of these technologies that implement them.

4.4.1 Atomicity, Consistency, Isolation and Durability

Atomicity, Consistency, Isolation and Durability (ACID) properties ensure consistency between database instances in the event of concurrent access and potential system failures. ACID properties for transactions in RDBMSs have been discussed in Chapter 3 – Table 4.1 provides a brief summary of these properties.

Table 4.1: Summary of ACID properties for transactions.

PROPERTY	DESCRIPTION
Atomicity	All transactional operations will complete or fail.
Consistency	None of the integrity constraints in a database will be violated.
Isolation	Each transaction is viewed as the only operation performed on the database.
Durability	Once a transaction is committed, that transaction cannot be reversed.

4.4.2 Consistency, Availability and Partition Tolerance (CAP) Theorem

The Consistency, Availability and Partition Tolerance (CAP) theorem was introduced by Brewer (2000) and proven by Gilbert and Lynch (2002). This theorem on distributed systems states that it is impossible to achieve all three of the properties – namely consistency, availability and partition tolerance – at the same time. Therefore only two of these properties can be satisfied at once (Gilbert & Lynch, 2002:58). Since most NoSQL DBMSs are distributed, the CAP theorem has important implications for the design of distributed systems. It is necessary to understand the different properties of the CAP theorem as individual DBMSs satisfy different properties (Oliveira da Silva, 2011:5). The definitions of these properties are given below (Pritchett, 2008:50; Pokorny, 2011:279):

- Consistency within this theorem is seen as a set of operations occurring at once. Once data are written to the database, everyone who reads from the database will see the most recent version of the data. The goal of consistency is to have commonly supported all-or-nothing semantics to allow low multisite transactions (Stonebraker, 2010a:8).
- Availability means that every operation must terminate with an intended response. To improve this intended response for every operation, many servers are used to act as a single database. The data are shared across a number of database nodes and data replications. This is known as high availability. The goal of availability is that the DBMS is online (Stonebraker, 2010a:8).

- Partition tolerance refers to the ability of the operation to complete even if individual components are unavailable. Thus, when parts of the database are completely inaccessible, data may still be read from and written to the database. By redirecting data writes designated to a particular node that is unreachable to nodes that are accessible, the unreachable nodes may receive all the missed data writes when it comes back online. The goal of partition tolerance is that the DBMS will still continue processing after parts have failed (Stonebraker, 2010a:8).

Stonebraker (2010a:8) explains that “in the NoSQL community, the CAP theorem has been used as the justification for giving up consistency.” This is supported by Helland (2011:43) who states that consistency in semantics of data is influenced by huge scales and transactions across all data are not supported by most large systems in the NoSQL genre.

RDBMSs favour high consistency to support reliability of data, whereas most NoSQL DBMSs relax consistency to gain availability.

Figure 4.19 illustrates the combinations of consistency (C), availability (A) and partition tolerance (P).

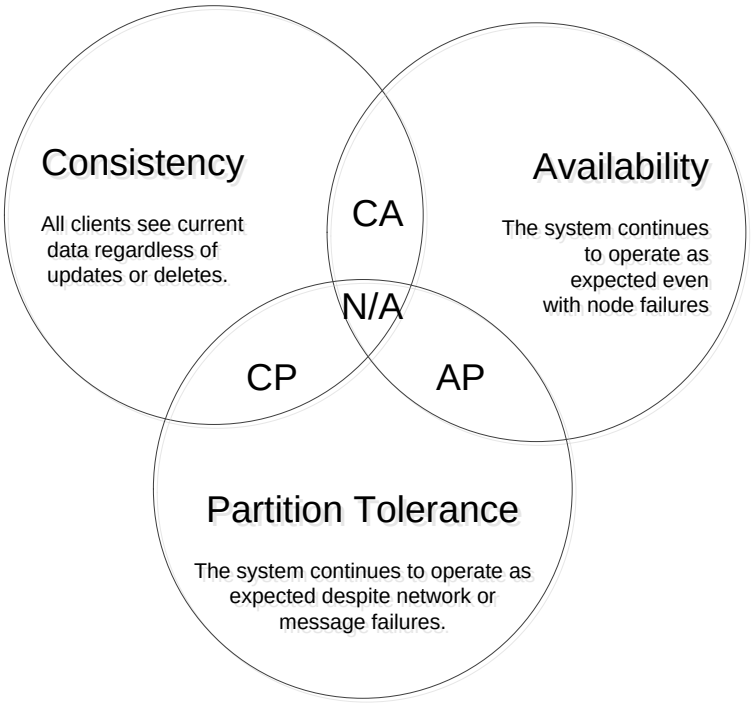


Figure 4.19: CAP theorem combinations of NoSQL technologies (Han et al., 2011:364; Moniruzzaman & Hossain, 2013:3)

4.4.3 Basically available, soft state and eventually consistent

The idea of basically available, soft state and eventually consistent (BASE) is that during transactions, updates are eventually propagated, with limited guarantees as to the consistency of reads (Cattell, 2011:12). ACID is a pessimistic approach and forces consistency at the end of every transaction. BASE is an optimistic approach and accepts that the database consistency will be in a state of eventual consistency (Pritchett, 2008:51). With the lack of ACID constraints, higher performance and scalability may be achieved (Cattell, 2011:12). BASE's availability property is achieved through supporting partial failures but not complete system failure. BASE is intrinsically connected with the CAP theorem and was proposed by Brewer in 2000 (Oliveira da Silva, 2011:5). It should be noted that not all NoSQL is full ACID or BASE compliant and is not suited for bank-like applications, because consistency is the most important feature in these applications (Hecht & Jablonski, 2011:341).

Eventual consistency is similar to consistency, but is a weaker form (Vogels, 2009:42). If no new updates are made to a record in a system that is distributed, provision is made for eventual consistency. This guarantees that the system will eventually be consistent and all data read will return the latest updated value. The inconsistency window is a timeframe from when the moment the data were modified to the moment the data are data consistent. A number of factors, including system loads, as well as the number of replications that need to take place, determine the maximum size of the inconsistency window (Oliveira da Silva, 2011:8). It should be noted that variations of eventual consistency models exist (Vogels, 2009:42), which are summarised in Table 4.2.

Table 4.2: Variations of eventual consistency quoted from Vogels (2009:42)

MODEL	DESCRIPTION
Causal consistency	If a certain process communicates to another process that it has updated a value, subsequent access by that process will return the updated value and a write is guaranteed to supersede the earlier write. Access by another distinct process that has no causal relationship to the originator process is subject to the normal eventual consistency rules.
Read Your Writes consistency	The effect of a write operation by a process on a variable is always seen in a subsequent read operation of the same variable by the same process.
Session consistency	A variation of read your writes, which ensures that a process is able to read its own writes during a specific session. From the context of that session the same process might see older values.
Monotonic reads	If a process reads the value of a variable, any subsequent reading of that variable by the same process should return the same value or a more up-to-date value.
Monotonic writes	A write operation of a variable by a process completes before the following write operation of the same variable by the same process. The system guarantees to serialise writes performed by the same process.

4.4.4 Scalability: vertical scaling vs. horizontal scaling vs. sharding

This section describes the different scalability architectures and then explains certain concepts such as shared nothing and the gossip protocol that aids in scalability.

The scalability architecture is defined by Bondi (2000:195) as “a desirable attribute of a network, system, or process”. Scalability was mainly divided into vertical scaling and horizontal scaling, but later sharding was introduced.

Vertical scaling (referred to as scale-up) invests in new and bigger servers (Pokorny, 2011:278) to gain a greater increase in performance to handle the workload. Information Systems (ISs) have relied on vertical scaling to improve performance for years (Pokorny, 2011:278). For example, when more users were added to the system, the server was upgraded or replaced by a bigger server. Figure 4.20 illustrates vertical scaling. The flaw of vertical scaling was that it did not guarantee reliability (Pokorny, 2011:278). If the single machine failed, the system failed and reliability became a flaw.

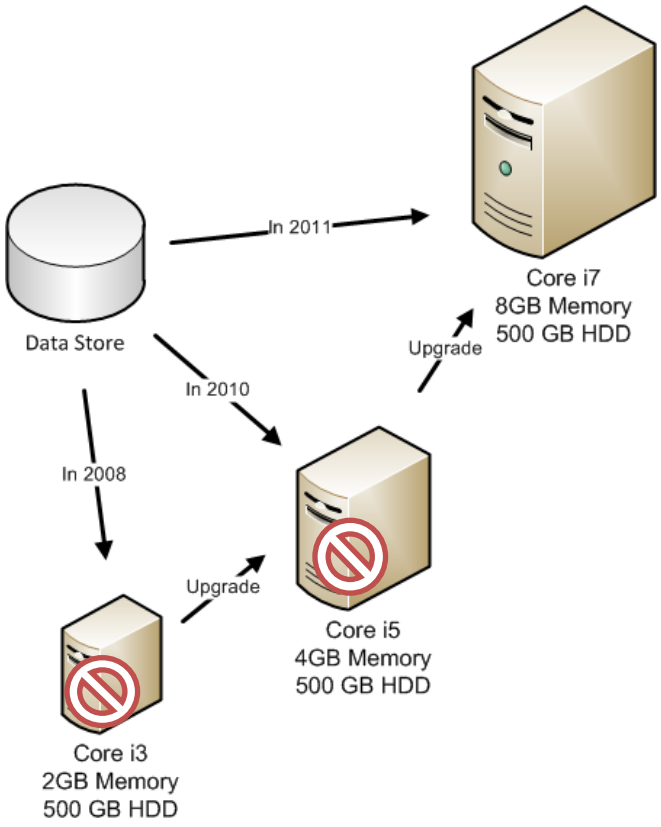


Figure 4.20: Vertical scaling example

In Figure 4.20 the machines are upgraded to support the database load. The previous machines are discarded.

The reliability flaw brought forth horizontal scalability (referred to as scale-out (Pokorny, 2011:278)) where data and processing loads were partitioned across multiple cheap servers (Cattell, 2011:13). For example, when more users were added to the system, more servers were added to handle the work load. Horizontal scalability ensured that a cheaper and more effective way of scaling was possible (Pokorny, 2011:278). Figure 4.21 illustrates horizontal scaling.

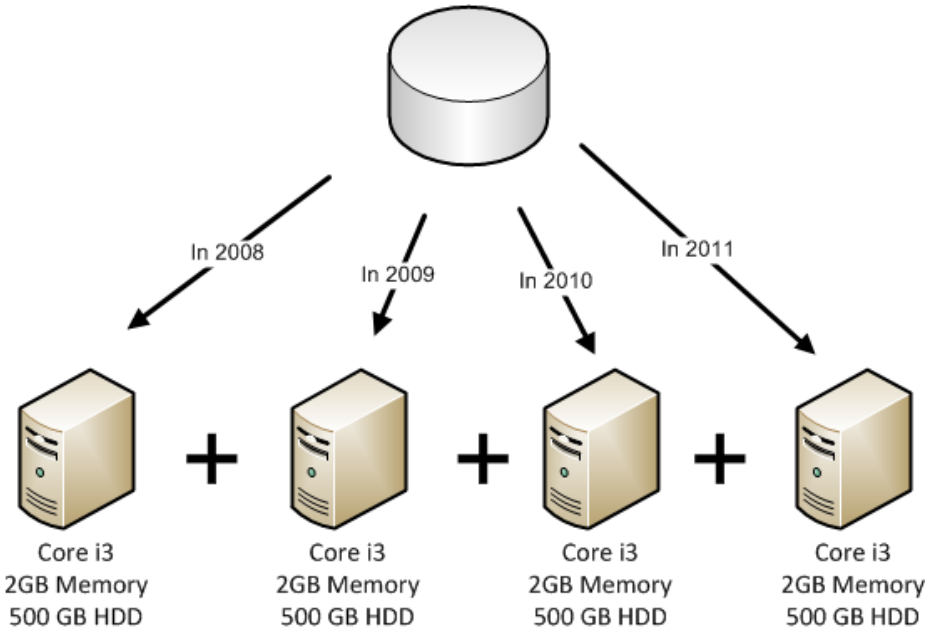


Figure 4.21: Horizontal scaling example

In Figure 4.21 the same commodity machine is added to the system to support the database load. Previous machines are not discarded but used throughout the cluster.

NoSQL databases had to relax some of the usual characteristics of RDBMSs to accomplish horizontal scalability (Pokorny, 2011:278). Some horizontal scalability is achieved with Simple DB, a document store, if the latest version is not required.

Horizontal scaling is also found in data warehousing (DW) database systems, but it is different because (Cattell, 2011:14):

- DW systems have complex queries that make use of joins to draw information from various tables; and
- DW systems have a very high read-to-write ratio – they are mostly read-only or read-mostly.

Sharding is a new architecture in scalability where data are split up and stored across multiple servers (Rahul, 2008; Ries, 2009) and there is a method to ensure access to data in the right place (Ries, 2009). In sharding, for example, a certain chunk of data of a company may be stored on one server while the rest of the data may be stored on a different server.

Sharding is known to be a federated model (Rahul, 2008), but Cattell (2011:13) states that in NoSQL there is no 'federated database' concept and that "each database is an island unto its own". This means that each machine found within the cluster is administered separately but still acts as one database (Cattell, 2011:13).

Project Voldemort, Riak and Tokyo Cabinet (which are all key-value stores) and SimpleDB, CouchDB and MongoDB (which are all document stores) implement sharding to some extend (Cattell, 2011). VoltDB, Clustrix, ScaleDB and ScaleBase which are RDBMS stores also implement sharding to some extend (Cattell, 2011).

Figure 4.22 is an example of sharding: the database is split into smaller pieces and each data piece is stored on a separate machine.

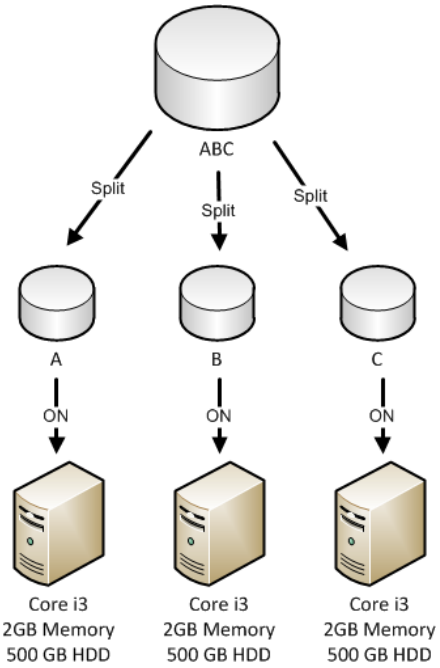


Figure 4.22: Sharding example

Scalability implementation differs between the different document stores, but a common characteristic found between document stores is that all the documents in document stores are always distributed over all the nodes in the system (Cattell, 2011:19).

Scalability is sometimes combined or used in conjunction with the shared nothing architecture and the gossip protocol that is described in the next subsections.

4.4.4.1 Shared nothing

Shared nothing (SN) is a strategy commonly found in the architecture of multiprocessor high-transaction-rate systems (Stonebraker, 1986:4). Shared nothing is where neither memory nor storage is shared among processors (Stonebraker, 1986:4). It is often presented between shared memory (SM) and shared disk (SD) architectures in the literature. Shared memory is where a central memory unit is used by multiple processors (Stonebraker, 1986:4). Shared disc is where a common collection of discs is shared between multiple processors with each processor having its own memory (Stonebraker, 1986:4).

Shared nothing is sometimes combined with horizontal scalability to provide, as Cattell (2011:12) describes it, ‘shared nothing’ horizontal scalability. A large amount of simple read and simple write operations are then supported by NoSQL systems (Cattell, 2011:12). Each node in the shared nothing horizontal scalability cluster thereby acts on its own.

4.4.4.2 Gossip protocol

The gossip protocol is a communication protocol that aids scalability and fault-tolerance in distributed systems (Allavena *et al.*, 2005:292; Bakhshi *et al.*, 2009:247). The gossip protocol works similarly to social networks (Allavena *et al.*, 2005:292). Each of the nodes in the cluster sends a message to a subset of the cluster, picked at random (Allavena *et al.*, 2005:292). These messages are repeatedly broadcasted at short intervals and the result is a higher probability of guaranteed delivery to all the nodes in the cluster (Allavena *et al.*, 2005:292). The following list demonstrates the operations of a generic gossip protocol (Bakhshi *et al.*, 2009:247):

- each node of the membership has some data associated with it;
- also, each of these periodically gossips those data with another node;
- node A, for example, randomly selects a node B, for example from a list of known nodes;
- node A sends a data-containing message to node B;
- node B sends back a response with a data-containing message; and lastly
- node A and node B update their data set by merging it with the received data.

In the case of Riak, the gossip protocol is used to track who is alive and who has what data (Cattell, 2011:15). There is no master node that tracks the status of the system and any client can be serviced by any node (Cattell, 2011:15).

4.4.5 MapReduce

MapReduce is a programming strategy that uses multiple computers to perform calculations over large data sets to enable parallel processing (Dean & Ghemawat, 2008:137). MapReduce was first presented by Jeffery Dean and Sanjey Ghemawat,

two employees of Google, in their paper “MapReduce: Simplified Data Processing on Large Clusters” (Dean & Ghemawat, 2008). Google developed it to support their indexing system (Dean & Ghemawat, 2008:137). The discussion of MapReduce is necessary because a number of NoSQL technologies make use of it.

At times it may be required to process a given big data set and to perform operations on the data set. At the same time, it may be that a single machine’s computing power is not enough, and then it may become necessary to distribute the tasks across several machines. The MapReduce programming strategy provides the opportunity for the application to automatically develop and distribute across multiple computers (Dean & Ghemawat, 2008:137). With this opportunity, the developer needs not be concerned about issues such as parallelisation, synchronisation and distribution (Oliveira da Silva, 2011:10).

The MapReduce programming strategy presents a two-step process: the ‘map function’ and the ‘reduce function’ (Oliveira da Silva, 2011:10). The map function processes a subset of the original set and returns the result. The master node uses divide-and-conquer techniques on the given problem, breaking it down into smaller chunks and distributing it across the child or worker nodes. The child or worker nodes then process the chunks and report an answer back to the master node (Oliveira da Silva, 2011:10). Once these worker nodes have reported an answer back to the master node, the reduce functions come into play. The master nodes receive all these answers and combine them to present them as output. This is stated as the final answer to the original problem (Oliveira da Silva, 2011:10). The MapReduce phases are represented in Figure 4.23.

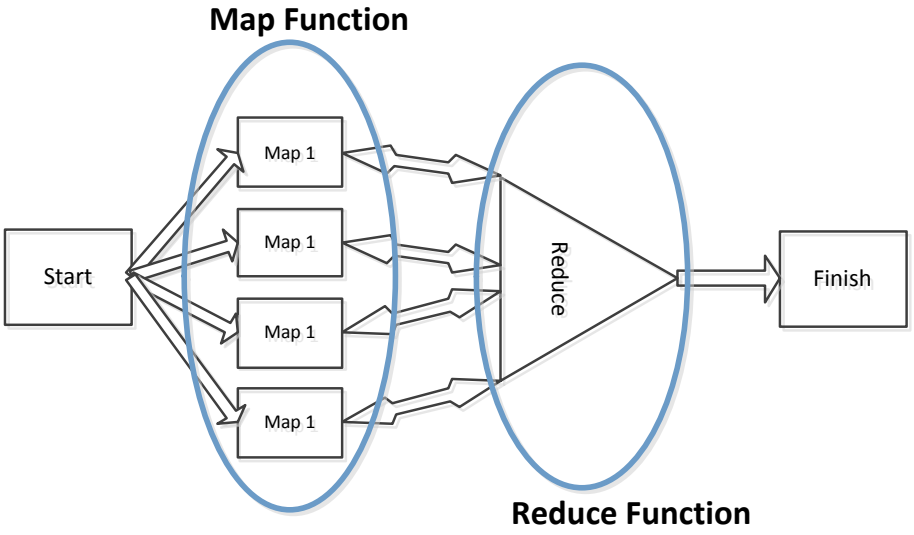


Figure 4.23: MapReduce graphical representation

In Figure 4.23, the map function takes the input of a pair of values and returns a set of intermediate key/value pairs. The reduce function accepts the intermediate key and a set of values for that key, to be merged to form a smaller set of values. Only one output value (which might be zero value), is to be produced per reduce invocation (Dean & Ghemawat, 2008:138).

Within a large set of documents, MapReduce may be a useful technique for distributing a task such as counting the number of occurrences for each word. Figure 4.24 presents an example of MapReduce where the number of occurrences for each word is counted (Dean & Ghemawat, 2008:138).

```
1   map (string key, string value):
2           //key: document name
3           //value: document contents
4           for each word w in value:
5                   EmitIntermediate (w,"l");
6
7   reduce (string key, Iterator values):
8           //key: a word
9           //values: a list of counts
10          Int result = 0;
11          for each v in values:
12                  Result += ParseInt(v);
13
14          Emit(AsString(result))
```

Figure 4.24: Code example of MapReduce (Dean & Ghemawat, 2008:138)

In the map function (starting at line 1) of Figure 4.24, the key/value pair is distributed for each word (lines 4 and 5). Here the key represents the word and the value represents the number of occurrences for that word. In the reduce function, all the results of distributed values (lines 7 to 12) are summed together (line 12) for a word. Then a result is returned (line 14) (Oliveira da Silva, 2011:11).

MapReduce is available and used in many NoSQL databases, such as Apache Hadoop. It was first implemented as proprietary software by Google (Oliveira da Silva, 2011:11). In addition to the use of MapReduce in NoSQL databases, it has also been adopted by many programming languages like Python, frameworks such as Apache Hadoop and JavaScript toolkits such as Dojo (Strauch *et al.*, 2011:50).

NoSQL databases makes use of MapReduce operations to perform aggregated operations similar to that of SQL COUNT, SUM, etc. (Oliveira da Silva, 2011:11). These operations are required with the distribution of extremely large data sets.

4.4.6 Concurrency control

It is necessary to implement strategies, such as locking, to avoid inconsistencies in parallel systems when many users need to access the same data (Hecht & Jablonski, 2011:339). Locking or a lock in the concurrency control of database systems refers to the guarantee of exclusive data item use in a current transaction (Morris *et al.*, 2013:580). When a lock is applied to data with certain exclusive access rights, no other processes trying to access the data will be allowed and will have to wait (Oliveira da Silva, 2011:9). Exclusive access is used in RDBMSs as a pessimistic approach (Hecht & Jablonski, 2011:339). Locks are suitable if the datasets are not blocked for a long time, but in database clusters, locks in particular become expensive and web applications need high read requests (Hecht & Jablonski, 2011:339). Massive performance loss is caused with these pessimistic consistency strategies of locks (Hecht & Jablonski, 2011:339).

Since there is a drawback with these pessimistic locking strategies, multi-version concurrency control (MVCC) and optimistic locking strategies have been developed.

4.4.6.1 Multi-version Concurrency Control (MVCC)

Multi-version concurrency control (MVCC) is a mechanism that enables concurrent access to a certain resource, such as a database (Bernstein & Goodman, 1983:482; Faller, 2009). It was first described in 1978 by David Reed in a dissertation on “Naming and Synchronization in a Decentralized Computer System” (Reed, 1978). Since 1978, MVCC was widely implemented in relational databases and also some NoSQL databases. The MVCC mechanism is an alternative to locking.

MVCC has the ability to support simultaneous access to data and also prevent data corruption and deadlocks (Oliveira da Silva, 2011:9). Exclusive access to resources

is not given under MVCC. Versions of the resources are rather kept, and when processes request data, the latest version of the data is presented. Therefore data may be read in parallel by processes, while other processes simultaneously write to the resource. This is an optimistic approach (Oliveira da Silva, 2011:9). To assist performance, MVCC relaxes strict consistency (Hecht & Jablonski, 2011:339). Wang *et al.* (2000:608) describe strict consistency as follows: “Whenever the user performs an update operation, the update is applied to all reachable copies as part of the update protocol”. Timestamps or revision numbers are attached to each data item, thereby achieving consistency (Oliveira da Silva, 2011:9). Concurrent access is managed by means of unmodifiable chronologically ordered versions (Hecht & Jablonski, 2011:339).

Table 4.2 in Section 4.5 lists some of the NoSQL DBMSs technologies that implement MVCC.

4.4.6.2 Optimistic locking

Optimistic locking is supported by many NoSQL DBMSs to support transactions without holding the dataset in exclusive access (Hecht & Jablonski, 2011:339). Each transaction checks any other transaction for data changes before committing the data (Hecht & Jablonski, 2011:339). If there are conflicts, then the transaction is rolled back (Cattell, 2011:15; Hecht & Jablonski, 2011:339). Optimistic locking functions well when changes to data are low. The rollbacks are cheaper than trying to lock datasets for exclusive access (Hecht & Jablonski, 2011:339).

4.4.7 Conclusion on distributed NoSQL databases

The CAP theorem states that only two of the three properties (consistency, availability and partition tolerance) can be satisfied at any given time. This theorem is the basis for all NoSQL DBMSs, which all incorporate two of these properties in some way or another.

BASE is an optimistic approach compared to ACID, and accepts that the database will eventually become consistent, but remains available to the users to use.

Scalability (vertical, horizontal or sharding) is implemented to allow databases to be distributed over many computers.

MapReduce enables parallel processing for large data sets. It enables program execution to become distributed across several machines without the programmer having to worry about the implementations.

MVCC, one of the concurrency control strategies, enable simultaneous access to a database. It prevents data corruption and deadlocks by locking the data and not allowing any other process to access the data.

Optimistic locking, another concurrency control strategy, checks for conflicts and only rolls back the data when conflicts are found.

Although not all these characteristics are implemented by all the technologies, they form the foundation for most NoSQL DBMSs. Table 4.3 at the end of this chapter lists some of the key NoSQL technologies and their implementation of these characteristics.

4.5 NOSQL TECHNOLOGIES

The development of NoSQL technologies has been in the news for the last few years, especially for web leaders such as Facebook, Twitter, Digg, Amazon, LinkedIn and Google (Moniruzzaman & Hossain, 2013:10). All these companies incorporate the technology in one form or another (Moniruzzaman & Hossain, 2013:10).

NoSQL has become a serious consideration for companies with massive data storage needs (Moniruzzaman & Hossain, 2013:11). Organisations that store massive amounts of unstructured data have turned to these non-relational databases (Moniruzzaman & Hossain, 2013:11).

Although Edlich (2009) states that more than 150 NoSQL databases currently exist, this number will be out of date by the time that this study is published. Therefore it is not possible to investigate them all. Figure 4.25 lists the most popular NoSQL databases, based on LinkedIn profile mentions. According to The-451-Group (2013), MongoDB has been the most popular NoSQL DBMS with a 45% share in the market as illustrated in Figure 4.25. MongoDB is followed by Cassandra in second place and Redis in third place.

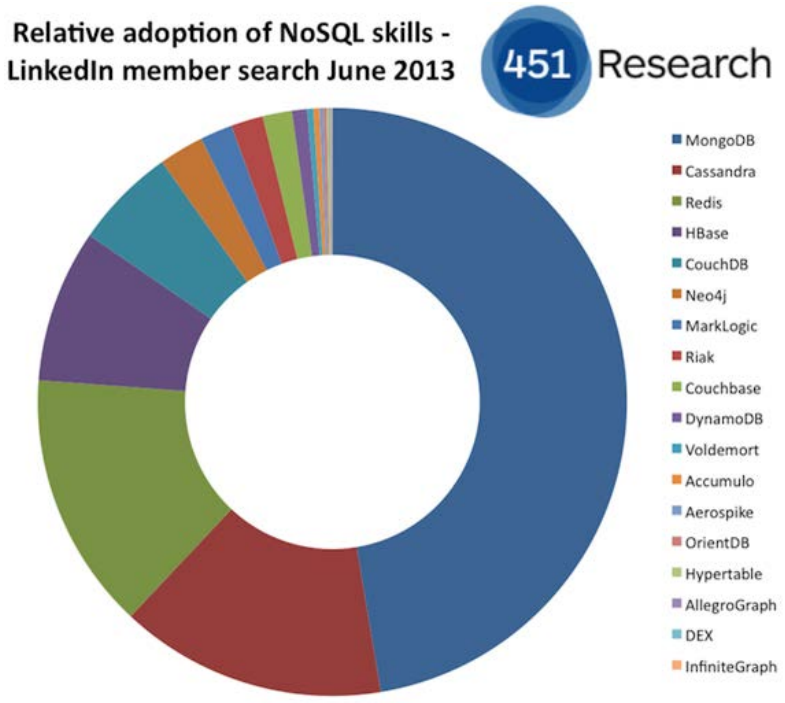


Figure 4.25: NoSQL LinkedIn skills index (The-451-Group, 2013)

Table 4.3 is a summary table generated from different sources to illustrate some of the NoSQL technologies and their implementations of data models, CAP properties, MapReduce and MVCC.

Table 4.3: Summary table compiled of characteristics implemented by some NoSQL technologies from different sources (Lith & Mattsson, 2010:24; Hecht & Jablonski, 2011:339; Padhy et al., 2011:19)

TYPE	DATABASE TECHNOLOGY	CAP PROPERTIES ADAPTED	CAN IMPLEMENT MAPREDUCE?	CAN IMPLEMENT MVCC?
Key-Value	Voldemort	AP	No	No
	Redis	CP	No	No
	Membase	Not available	No	No
	Riak	CP	Yes	Yes
Document	MongoDB	CP	Yes	No
	CouchDB	AP	Yes	Yes
Column Oriented	Cassandra	AP	Yes	No
	HBase	CP	Yes	No
	Hypertable	Not available	Yes	Yes
Graph Databases	Sesame	Not available	No	No
	BigData	Not available	No	No
	Neo4J	Not available	No	No
	GraphDB	Not available	No	Yes
	FlockDB	Not available	No	No

MongoDB is the database used in this study as it is the most popular, contains already built libraries for C#.net and VB.net, and it is easy to set up (MongoDB, 2007).

4.5.1 MongoDB

MongoDB is an open-source document data store database that stores its data in binary JSON-style documents (Lith & Mattsson, 2010:24), referred to as BSON. BSON is a binary-encoded serialisation of JSON-like documents (MongoDB, 2007)

and is an open standard. The name MongoDB was derived from the adjective “humongous” (Membrey *et al.*, 2010:3). Humongous means extremely large, huge or enormous (Oxford Dictionaries & Press, 2013). Since the MongoDB name was derived from the word humongous, it may be seen as a scalable, high-performance, open-source NoSQL database. It was written in C++ (MongoDB, 2007; Cattell, 2011:17). A company called 10gen Inc. started the developing of MongoDB (Cattell, 2011:17) in 2007. 10gen Inc. offers professional services to support the database (Strauch *et al.*, 2011:76).

The main purpose of MongoDB is to store data from applications that do not require strong transactional requirements. Similar to other document store databases, it is unstructured, user friendly, and easily scalable on different servers (Lith & Mattsson, 2010:25). MongoDB became a stand-alone and open-source product with an AGPL licence in 2009 (MongoDB, 2009). The key features of MongoDB are listed as follows (Membrey *et al.*, 2010:11-16):

- document-oriented storage;
- supporting dynamic queries;
- indexing documents;
- leveraging geospatial indexes;
- profiling queries;
- updating information in-place;
- storing binary data;
- replicating data;
- implementing auto sharding; and
- using map and reduce functions.

MongoDB has some key differences compared to the other document data stores. These differences are (Cattell, 2011:18):

- the support of automatic sharing;
- replication is used for failover and not scalability;
- dynamic queries are supported, which is listed as a feature of MongoDB; and

- provision for atomic operations on fields.

Consistency and partition tolerance are the CAP theorem properties employed by MongoDB. Neither MVCC nor optimistic locking is used as a scalability feature for concurrency control in MongoDB. MongoDB rather employs a last-update-wins strategy for the modification of data (Hecht & Jablonski, 2011:339). MongoDB does employ MapReduce as a programming strategy to distribute the workload across the cluster. Failover and recovery are automatically supported in MongoDB by master/slave replication (Cattell, 2011:18). GridFS for large binary objects such as images and videos are also supported in MongoDB.

4.6 LOOKING BEYOND TOWARDS NEWSQL

Most of the literature reviewed in this study concerned NoSQL, but the study also needs to take into account new movements in the DBMS world such as NewSQL and Entity-Attribute-Value (EAV).

NewSQL refers to a new era of RDBMS that could accommodate the same performance in scalability similar to those found in NoSQL and still incorporate the classic ACID properties (Moniruzzaman & Hossain, 2013). NewSQL may also be referred to as scalable relational systems (Cattell, 2011:20). The term was coined by The-451-Group in 2001 in their article: “How will the database incumbents respond to NoSQL and NewSQL?” (Aslett, 2011). The article focused on how emerging database systems may be challengers to established database systems.

NewSQL will still support the common relational data model and make use of SQL as a query language, but architectures may differ among one another (Cattell, 2011:20).

H-Store is one of the first known NewSQL DBMSs (Aslett, 2008; Monash, 2008). Other emerging NewSQL DBMSs include MySQL Cluster, VoltDB, Clustrix, ScaleDB, ScaleBase and NimbusDB (Cattell, 2011:20).

Building further on the movement of NewSQL are the Entity-Attribute-Value (EAV) data models. EAV refers to a data model that allows the dynamic creation of entities' attributes (Holscher *et al.*, 2013). The creation of an attribute table and the storing of attributes in this table in the EAV model supports variable attributes (Holscher *et al.*, 2013). The Object–Attribute–Value model, the vertical database model and open schema are known associations for EAVs.

NewSQL and EAV fall beyond the scope of the study but it is important to take note of them. This is because although RDBMSs have been catching up on scalability, they are still limited to a structured data model. This limitation still offers the gap for NoSQL to capture any data in any format without the risk of data loss.

As these technologies gain acceptance, a need may develop to transform NoSQL data to structured data. This need may be addressed by the artefact developed in this study.

4.7 CONCLUSION

The objective of this chapter was to gain an understanding of NoSQL data stores and NoSQL technologies. The focus was on the document data stores implemented by some of these technologies. The objective was met by defining NoSQL, describing the evolution of NoSQL, NoSQL data models, distributed NoSQL databases, and the NoSQL technology implemented in this study.

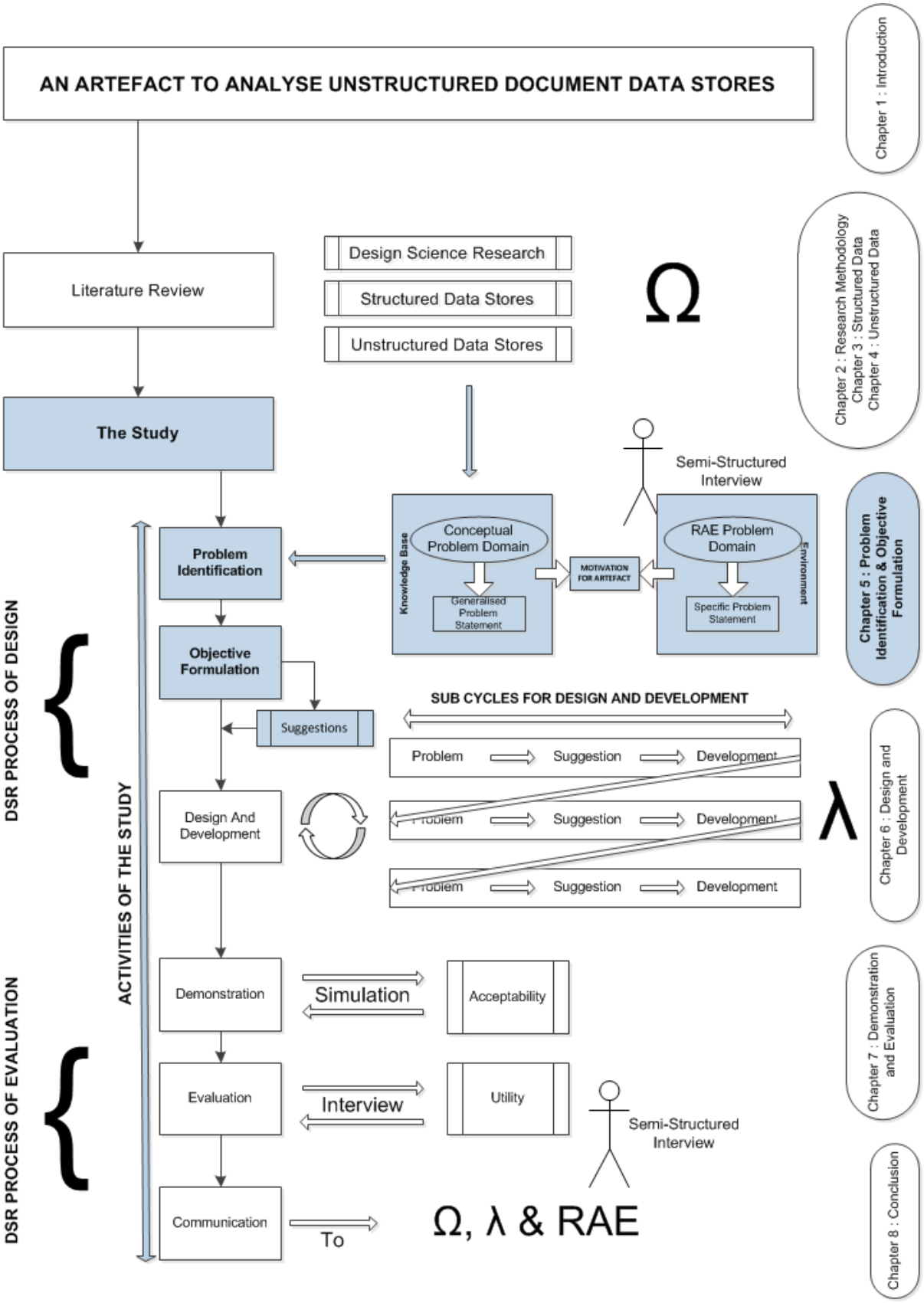
NoSQL is a non-relational, distributed, open-source and horizontally scalable DBMS that evolved from the need for better horizontal scalability regarding the distribution of data over multiple servers. It also addresses the need to store unstructured data which is represented by four data models: key-value stores, document stores, column-oriented stores and graph databases.

The data stores relevant to the study are document data stores. Document data stores contain key-value pairs and can contain sub-structures such as documents and lists.

MongoDB is a DBMS document store developed by 10gen Inc. and is used in this study. MongoDB uses binary-encoded serialisation of JSON-like documents for the storage of data.

An ideology that is propagated by the NoSQL community as stated by Hecht and Jablonski (2011:341) is: “Use the right tool for the job”.

The next chapter introduces the problem domain from a conceptual and real application environment (RAE) point of view. In addition, the objectives for the design and development of the artefact are introduced.


AN ARTEFACT TO ANALYSE UNSTRUCTURED DOCUMENT DATA STORES
Literature Review
The Study
Design Science Research
Structured Data Stores
Unstructured Data Stores
Ω
Semi-Structured Interview
Problem Identification
Objective Formulation
Suggestions
Design And Development
Demonstration
Evaluation
Communication
To
Knowledge Base
Conceptual Problem Domain
Generalised Problem Statement
MOTIVATION FOR ARTEFACT
RAE Problem Domain
Specific Problem Statement
Environment
SUB CYCLES FOR DESIGN AND DEVELOPMENT
Problem
Suggestion
Development
λ
Simulation
Acceptability
Interview
Utility
Semi-Structured Interview
Ω, λ & RAE
DSR PROCESS OF DESIGN
DSR PROCESS OF EVALUATION
ACTIVITIES OF THE STUDY
Chapter 1 : Introduction
Chapter 2 : Research Methodology
Chapter 3 : Structured Data
Chapter 4 : Unstructured Data
Chapter 5 : Problem Identification & Objective Formulation
Chapter 6 : Design and Development
Chapter 7 : Demonstration and Evaluation
Chapter 8 : Conclusion

CHAPTER FIVE: PROBLEM AND OBJECTIVES

5.1 INTRODUCTION

The primary objective of this study is to develop an artefact which analyses document data stores in terms of structured data model constructs. In order to achieve this, this study formulated a design science research (DSR) methodology in Chapter 2. This chapter presents the first two activities of this study's DSR methodology – problem identification and objective formulation.

A problem statement in the context of Information Systems (IS) is an assertion and classification of problems, opportunities and directives (Bentley & Whitten, 2007:82). Deriving objectives may assist in solving a problem. An objective is an expectation to achieve something within sufficient boundaries, therefore it may be seen as a measurement of achievement (Bentley & Whitten, 2007:182).

The first objective of this chapter is to investigate the problem domain from a conceptual view and from the actual real application environment (RAE). The second objective of this chapter is to formulate the objective of the artefact by using knowledge from both the problem domains and the descriptive knowledge from Chapters 3 and 4. The last objective of this chapter is to make suggestions to achieve the objective of the artefact.

The discussion of this chapter is divided into the following sections: problem identification (Section 5.2); objective formulation (Section 5.3); suggestions to achieve the objective of the artefact (Section 5.4); and finally, the conclusion (Section 5.5).

5.2 PROBLEM IDENTIFICATION

The problem domain for this DSR study is obtained from both a conceptual viewpoint and the RAE.

Firstly, the conceptual problem domain is investigated and analysed to formulate a general problem statement. The resulting general problem statement is supported by the descriptive knowledge discussed in Chapter 4. The left block of Figure 5.1 demonstrates the general problem domain and statement.

Secondly, the RAE is discussed and analysed to formulate a specific problem statement for the specific RAE problem domain. The resulting specific problem statement is supported by an interview conducted with a representative in the RAE. The right block of Figure 5.1 demonstrates the RAE problem domain and the specific problem statement.

Finally, these problem domains provide substantial evidence to motivate the development of the artefact, as demonstrated in Figure 5.1.

Figure 5.1 closely relates to Figure 2.4, the IS Research Framework, in Chapter 2. The difference between the two figures is the source of the problem. In Figure 2.4, the problem is usually drawn from the environment, and in Figure 5.1, the problem is drawn from both the environment and the knowledge base.

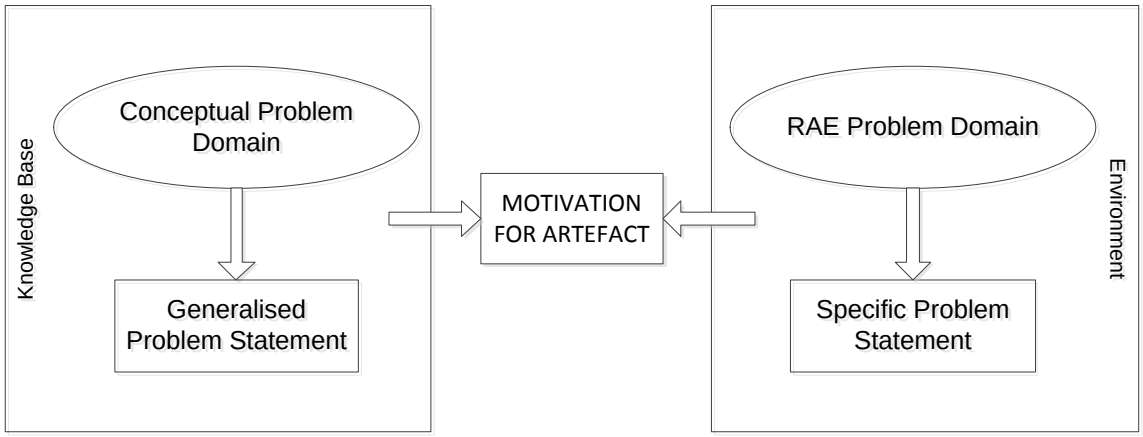


Figure 5.1: Initiation of the problem domain

The motivation for the artefact is discussed in Section 5.2.3 and the problem domains are discussed in the sections that follow.

5.2.1 Conceptual problem domain

Section 5.2.1.1 gives a summary of descriptive knowledge, and Section 5.2.1.2 discusses the formulation of a general problem statement that is derived from the descriptive knowledge.

5.2.1.1 Summary of unstructured descriptive knowledge

Table 5.1 is a summary of the descriptive knowledge of this study and it categorises the knowledge into themes. The first column is a coding value used to refer to in the development of the generalised problem statement. The second column is the theme identified for the descriptive knowledge. The third column is a quotation from the literature review concerning the descriptive knowledge. Finally, the fourth column is a reference to the applicable section of the literature review.

Table 5.1: Descriptive knowledge of this study.

CODE	THEME	DESCRIPTION	LITERATURE REVIEW SECTION
Lit:Data:Unstruct	Unstructured Data	• "next generation database management system (DBMS)" • "large amounts of web-generated data" • "non-relational, distributed, open-source and horizontally scalable" • "the ability to dynamically add new attributes to data records"	Section 4.1 Section 4.2
Lit:Stores:Unstruct	Unstructured Data Sources	• "Businesses, governments, public health organisations and social scientists" • "online buyers' purchases, web	Section 4.1

		searches, website interactions, blogs, tweets and product searches” • “Data are scattered across such a cluster” • “daily operations”	
Lit:Model:Key	Unstructured Data Models: Key-value Stores	• “program to store data in a schema-less or unstructured manner” • “isolated and independent and therefore relationships between data values of different keys may only be managed in programming logic” • “rich ad hoc queries and analytical features are omitted” • “accommodate any data structure possible for that specified key”	Section 4.3.2
Lit:Model:Docu	Unstructured Data Models: Document Data Stores	• “keep any pointer-less object” • “do not have any schema or structure restrictions” • “may contain and can easily add any attribute and additional attributes” • “the ability of supporting secondary indexes” • “it can handle multi-attribute lookups” • “desirable for flexible web applications” • “data integration and schema migration tasks”	Section 4.3.3
Lit:Model:Column	Unstructured Data Models: Column-oriented	• “relationships are not supported among the datasets” • “extra features also have to be managed with programming logic” • “can store multiple attributes per	Section 4.3.4

		identifier key”	
Lit:Model:Graph	Unstructured Data Models: Graph Databases	• “generally schema free” • “concerned with relations and the visual representation” • “suited for location-based services, finding common friends on social networks, finding the shortest path between two subjects, and knowledge representations”	Section 4.3.5
Lit:NEWSQL	NewSQL Movement	• “could accommodate the same performance in scalability similar to those found in NoSQL and still incorporate the classic ACID properties” • “still support the common relational data model and make use of SQL as a query language, but architectures may differ among one another”	Section 4.6

In the next section Table 5.1 is used to formulate a problem statement for the general problem domain. The coding is used in the text to refer back to the source of descriptive knowledge in the table.

5.2.1.2 Generalised problem statement

In this section, the descriptive knowledge from Table 5.1 is used to formulate a generalised problem statement.

It can be concluded from the descriptive knowledge given in Table 5.1 that unstructured data with the following characteristics exist:

1. consist of large amounts [Lit:Data:Unstruct];
2. are non-relational [Lit:Data:Unstruct];

3. are dynamic [Lit:Model:Key, Lit:Model:Docu, Lit:Model:Column, Lit:Model:Graph];
4. are drawn from various sources [Lit:Stores:Unstruct]; and
5. are placed in unstructured data models [Lit:Model:Key, Lit:Model:Docu, Lit:Model:Column, Lit:Model:Graph].

A general problem statement can be developed as:

> *Data exist from various sources [4] and in vast quantities [1] that are unstructured [5] and too complex [2, 3] to be viewed or interpreted by a human.*

The generalised problem statement is the first motivation for the development of the artefact. The developed artefact could benefit the viewing and interpretation of the data.

The focus of this study is to solve the stated problem to a certain extent. The problem will be solved for one specific type of unstructured data, namely document data stores. The document data store model is the most interesting data model, as it does not contain any schema or structure restrictions, and attributes may easily be added.

The next section explores the problem domain from the RAE point of view.

5.2.2 Real application environment problem domain

The section on the RAE problem domain is divided into: Section 5.2.2.1, the background of the RAE, Section 5.2.2.2, the analysis of the qualitative data, and Section 5.2.2.3, findings to formulate a specific problem statement based on these data.

5.2.2.1 Background of the real application environment

Since the practical application of the artefact in a real world environment is of utmost importance, a client environment was identified. Newcom Fluid Management (NFM) is used as the real application environment in this DSR project. To understand the requirements of the artefact, background information on NFM is given here.

NFM officially started operations in June 2011 as a company providing fluid management systems and services to the southern African market. The market segments include:

- agriculture;
- transport and commercial logistics organisations;
- industrial, construction and mining organisations; and
- marine organisations.

The shareholders who started NFM have all worked in the fluid management industry for several years and have accumulated extensive knowledge and experience in providing clients with effective products and services to manage their liquid resources.

NFM started operations to provide fluid management solutions to the southern African markets in an innovative and fresh way. Operations include:

- re-examination of conventional management systems;
- improved data management platforms;
- cost-effective systems; and
- flexible system acquisition models.

NFM has grown substantially since its inception. They currently have a wide footprint in the South African market for operational fluid management systems, stretching from the Western Cape to Limpopo. A wide variety of clients in different market segments are included in this footprint.

With the skills, resources and experience acquired by NFM, they are able to provide clients with cost-effective and high-quality fuel management solutions. NFM is also able to provide recommendations on refuel infrastructure, refuel cycles and other elements in the fuel cycle to assist clients to obtain the best possible fuel management solution.

5.2.2.2 Situational analysis of the RAE

A semi-structured interview was conducted with a representative of NFM. The purpose of the interview was to determine:

a) whether the company had access to unstructured data;
b) whether the company would benefit from the use of unstructured data; and
c) whether an application that supports the identification of unstructured data would be useful to the company.

The theme questions for the interview, along with the motivation for these questions, are listed in Table 5.2. The interview was transcribed and then analysed to identify possible requirements of the RAE. The transcription is given in Appendix B.

Table 5.2: Interview theme questions and their motivations

THEME QUESTIONS	MOTIVATION	THEME
Are the data collected differently and in different formats?	Find out if NFM has data in different formats in order to illustrate the need for different structures.	Data Collection Method and Types
What existing data are available?	Evaluate the manner in which the current data are stored. Describe the need to view data in a simple and readable format.	Structured Data Stores
Are the data stored in a structured manner?		Structured Data Stores
Is there hidden information?	Evaluate if it is possible that NFM may have unstructured data. Describe the need to	Unstructured Data

Is it possible for these data to be unstructured?	investigate for new information.	Unstructured Data
Is there a possibility for discarded data or unused data?	Evaluate if NFM loses valuable information. Describes the need to investigate for new information.	Unused data
Is there a way in the study to self-evaluate the use of the artefact?	Evaluate existing data to create self- and prior evaluation for the artefact.	Self-evaluation
Could this data be important or useful to the company?	Describe the importance to the company.	Company Importance
Future possibilities available?	Describe the importance to the company for future possibilities.	Future Possibilities

Table 5.3 provides the analysed data and extracted data based on the interview, using the qualitative data analysis technique discussed in Chapter 2. The first column presents the coding value used to refer to the analysis. The second column presents the questions from Table 5.2. The third column contains the quoted response from the RAE representative, Mr Oosthuizen (2013b), and finally the fourth column provides the theme that has been identified.

Table 5.3: Qualitative data analysis of the situation interview

CODE	THEME QUESTIONS	QUOTATION	THEME
Data:Diff	Are the data collected differently and in different formats?	"We have a very diverse system configuration and we work within all these configurations and connect within all these configurations' data". "The data will differ because we have different clients and different markets with different needs, whereby we capture different data for different clients and	Data Collection Method and Types

		communicate them in different mechanisms, and also the media available". "So yes, data differs in terms of input, and then also because of functionality and client dependence".	
Data:Cur:Aval	What existing data are available?	"There's a lot of space for data, and the primary sets that we look at are date and time, fuel quantities, reasons, people, users, locations, errors, abnormalities, and then probably a couple of other variables".	Structured Data Stores
Data:Cur:Aval	Are the data stored in a structured manner?	"The information on the field side from the raw industrial computers is in a raw text file with a fixed structure, which is then converted into a MySQL database".	Structured Data Stores
Data:Unstruc:Poss	Is there hidden information?	"I believe so, definitely." "There are a lot of complex relationships that can be derived from having a more elaborate data structure and looking at the relationships between different parameters. At this point, because we do not have all the parameters, or even more parameters, it's difficult to ascertain what relationships can or cannot exist". "If there are additional parameters it might be interesting to note, specifically in our applications, to see if we have trends of vehicle[s] who has high fuel usages, the basis of some other form of operation that they are currently doing, which parameter we do not capture currently".	Unstructured Data
Data:Unstruc:Poss	Is it possible for these data to be unstructured?	"…expand the data selection and capture more parameters we can actually deduce a lot more information".	Unstructured Data
Data:Unused	Is there a	"JaYes, at this point it's just discarded".	Discarded

	possibility for discarded data or unused data?		Data
Eval:Self	Is there a way in the study to self-evaluate?	"…the existing structures, what we have, so that's what I can use to measure existing attributes, but new attributes, it's going to be new territory for us as well". "…date and time, fuel quantities, reasons, people, users, locations, errors, abnormalities, and then probably a couple of other variables".	Self-evaluation
Busi:Importance	Could this data be important or useful to the company?	"Competitive edge". "…develop new algorithms for clients based on data captured that can predict a lot more accurately certain situations in terms of fuel usage, fuel monitoring, equipment, production, statistics, and we can actually add a lot more value to them by capturing a lot more information and giving them some feedback which was never even looked at." "It can open up the possibility of acquiring a lot more types of equipment, physical hardware that can be used, and also reducing the development time of the protocol integration and implementation". "Breaking new ground in our market can improve technology and create new opportunities which we are not even now aware of, so new business development is also a big possibility". "The goal of the company is to make money, and if I can add more value with my system than the competition's system can, then I will get the business and I can sustain the business by continuously giving my client valuable information, and I can	Business Importance

		only do that by capturing more data and deriving more relationships from what we currently have". "Yes, there is a lot of time and money spent on developing the protocol for each of our field types, and it necessarily means that we need to go to a development cycle on the software side when we include new hardware, because it's a- we need to integrate a new device with the new protocol to eventually get the data into our fixed database structure." "At this point I don't think so, just that to think of a possibility that I can capture data from devices, and somehow get them into the same database for analysis, and also on structured data in the sense that I can capture more than I think I could have captured, it can add so much value."	
Busi:Possi	Future possibilities available?	"…a possibility that I can capture data from devices, and somehow get them into the same database for analysis, and also on structured data in the sense that I can capture more than I think I could have captured, it can add so much value. It can open up the future for a lot of different industries, so it's a very, very innovative research topic for me".	Business Possibilities

5.2.2.3 Specialised problem statement

The company collects data that are diverse [Data:Diff]. The data are diverse because of the different clients, markets and media available [Data:Diff]. The data currently collected by NFM include: date and time, fuel quantities, reasons, people, users, locations, errors, etc. [Data:Cur:Aval]. The data are stored in a structured way [Data:Cur:Aval], but there are also data that are discarded [Data:Unused] due to their very structured format. NFM currently spends a lot of money on developing protocols

for each device, thereby trying to integrate their fixed database structure [Busi:Importance].

The company representative's response to the question: "Is there hidden information?" was, "I believe so. I believe so, definitely" [Data:Unstruc:Poss]. This indicates that there is a good possibility for unstructured data to be found among the data. By identifying the loss of additional parameters [Data:Unstruc:Poss, Data:Unused] and by expanding the capturing method [Data:Unstruc:Poss], the company expects to gain a competitive advantage [Busi:Importance]. Without the limitation of the protocol implementations, NFM could incorporate more physical equipment, adding value to their business [Busi:Importance]. Finally, more information may be generated and value may be added for their clients.

Capturing and analysing unstructured data may open up future possibilities for the company. NFM believes new industries may present new opportunities, such as new clients [Busi:Possi], allowing them to attain their main goal of increasing profits [Busi:Importance]. Some of these opportunities include acquiring new equipment and reducing development and integration time [Busi:Importance]. NFM foresees that this could lead to discovering unknown opportunities, and these opportunities may lead to their breaking into new markets and industries [Busi:Importance].

This study indicates that NFM has access to unstructured data that are hidden. By developing an artefact that could analyse these unstructured data, the company may benefit by getting access to the data and then utilising the currently hidden information.

Statements made by the RAE representative enabled the formulation of a specialised problem statement for the RAE problem domain. This statement incorporates some of the properties of the generalised problem statement to indicate applicability of the generalised problem domain.

The specialised problem statement is as follows:

> *NFM has vast amounts [Data:Unstruc:Poss] of data that are unstructured [Lit:Model:Docu, Data:Unstruc:Poss] and contain information that is hidden [Data:Unstruc:Poss] and not fully utilised [Data:Unused] by NFM to provide better services to their client base.*

The specialised problem statement is the second motivation for the development of the artefact. NFM believes that by capturing all the data in the same database for analysis and presenting a structured output, much more value than they had thought [Busi:Importance] could be added. The artefact developed could aid the RAE to fully utilise the data and thereby provide more and better services to the RAE client base.

The known attributes such as date and time, fuel quantities, reasons, people, users and locations [Eval:Self] provided some self-evaluation purposes prior to the actual evaluation of the artefact.

5.2.3 Motivation for the development of the artefact

The problems presented in Sections 5.2.1.2 and 5.2.2.3 drive the motivation for the development of the artefact. The artefact could solve the problems identified to an extent, create more research opportunities in the conceptual domain and provide more business opportunities for the RAE.

As explained in Chapter 4, RDBMSs are catching up on horizontal scalability – introducing a NewSQL movement [Lit:NEWSQL]. This NewSQL movement is the third motivation for the development of the artefact as companies may want to transform unstructured data into a structured format.

The artefact developed in this study will provide the groundwork for further progress in the field of analysis and transformation of structural data.

The next section introduces the objectives of the artefact.

5.3 OBJECTIVE FORMULATION

The objective formulation activity for this DSR study was accomplished using information from the problem domain. As described in Section 2.7, this study used the combined DSR methodology from Peffers *et al.* (2008) and Vaishnavi and Kuechler (2004). The second activity of Peffers *et al.* (2008) is object formulation, while the second activity of Vaishnavi and Kuechler (2004) is suggestion. This study incorporates the suggestion activity as a sub-activity of objective formulation activity from Peffers *et al.* (2008). The suggestions formulated are implemented in the design and development activity discussed in Chapter 6.

In order to address the problems presented in Sections 5.2.1.2 and 5.2.2.3 – to identify hidden data – an objective for the artefact needs to be formulated. This objective is formulated using the descriptive knowledge of structured stored data explored in Chapter 3.

The data to be viewed and interpreted are complex; however, a simple textual representation of the data will reduce this complexity. The relational data model described in Chapter 3 is a simple representation that can easily be viewed and interpreted. In order for the data to be viewed as relational, it needs to have entities, attributes and relationships.

Document stores described in Chapter 4 (Section 4.3.3) offer a unique way of storing data, which include an initial document, keys with values, sub-documents and arrays. This study proposes that the document store components be analysed and transformed into a relational data structure, which can be easily viewed and interpreted by humans.

Therefore this study presents the following objective for the artefact:

> *The artefact needs to analyse the document data store in order to identify entities, attributes and relationships.*

Figure 5.2 illustrates the objective of the artefact where the artefact needs to take data, process it and present output.

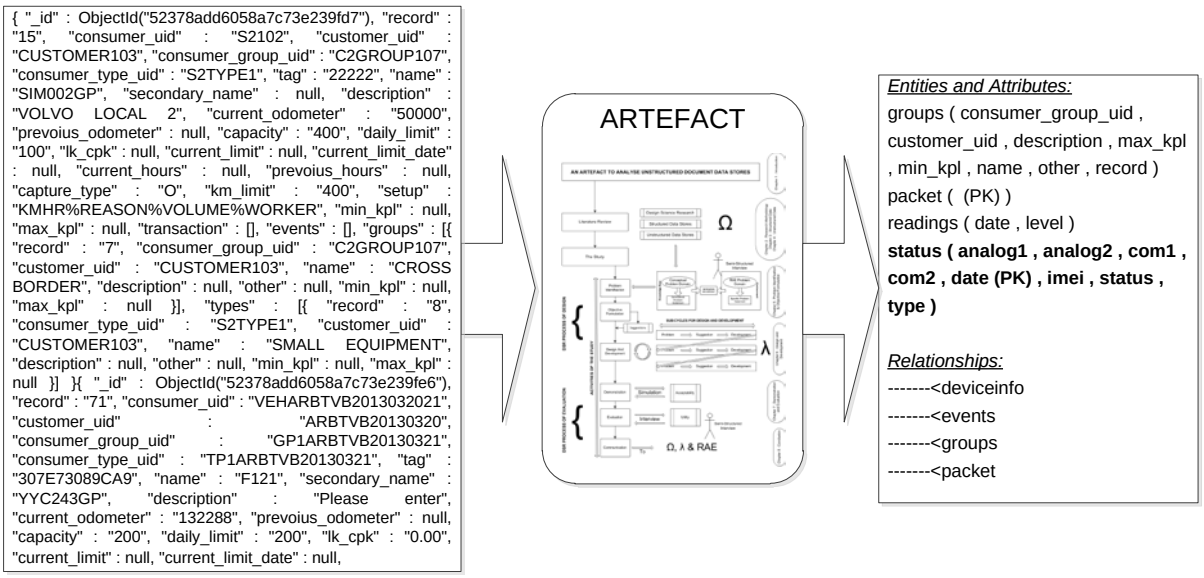


Figure 5.2: Processing of data by the artefact

5.4 SUGGESTIONS TO ACHIEVE THE OBJECTIVE OF THE ARTEFACT

Suggestions made by the researcher on how to identify entities, attributes and relationships in document stores are presented in Table 5.4

Table 5.4: Suggestions for the artefact as proposed by this study

#	SUGGESTIONS ARE THAT:
1	The initial document is identified as the first entity and possibly also the associative entity.

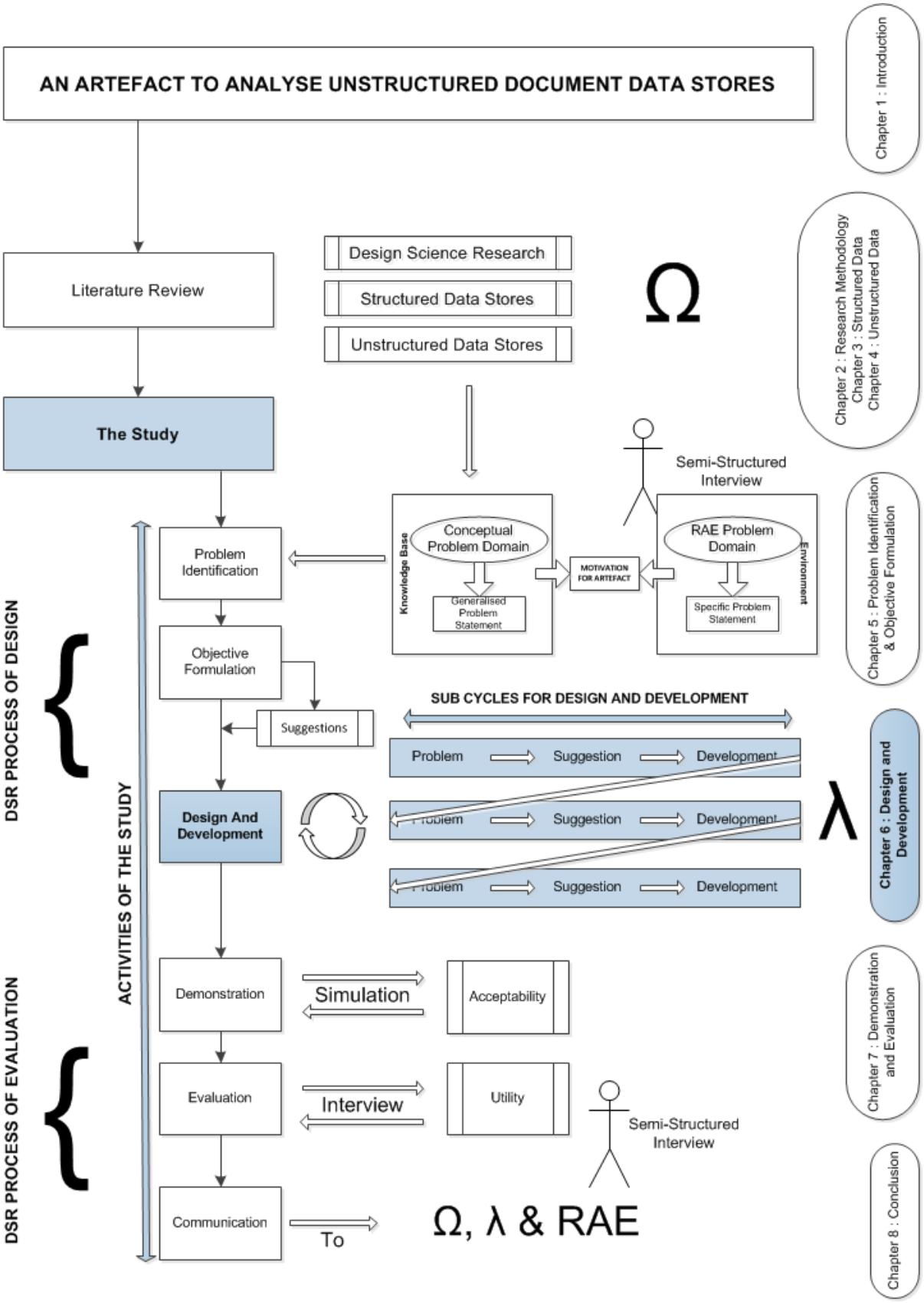
2	The keys are identified as attributes.
3	Sub-documents and arrays within the document are considered as entities, with attributes.
4	The attributes and their values are evaluated for uniqueness in order to identify possible primary keys.
5	Sub-documents and arrays within the parent document, already suggested as entities with attributes, resemble one-to-many relationships.

5.5 CHAPTER CONCLUSION

The objective of this chapter was to identify problem domains and formulate the objective for the artefact.

The problem was described from both a conceptual view and a RAE view. The problem faced by NFM is that they have vast amounts of data that are unstructured and contain hidden information. The development of an artefact could possibly assist NFM to identify useful hidden data, enabling them to provide better services to their customers.

The objective for this artefact is to identify entities, attributes and relationships. Finally, as a sub-activity of objective formulation, suggestions were made to support the objective for the artefact. The next chapter demonstrates the design and development of the artefact using these suggestions.


AN ARTEFACT TO ANALYSE UNSTRUCTURED DOCUMENT DATA STORES
Literature Review
The Study
Design Science Research
Structured Data Stores
Unstructured Data Stores
Ω
Semi-Structured Interview
Knowledge Base
Conceptual Problem Domain
Generalised Problem Statement
MOTIVATION FOR ARTEFACT
RAE Problem Domain
Specific Problem Statement
Environment
Problem Identification
Objective Formulation
Suggestions
SUB CYCLES FOR DESIGN AND DEVELOPMENT
Problem
Suggestion
Development
Design And Development
λ
DSR PROCESS OF DESIGN
ACTIVITIES OF THE STUDY
Demonstration
Simulation
Acceptability
Evaluation
Interview
Utility
Semi-Structured Interview
DSR PROCESS OF EVALUATION
Communication
To
Ω, λ & RAE
Chapter 1 : Introduction
Chapter 2 : Research Methodology
Chapter 3 : Structured Data
Chapter 4 : Unstructured Data
Chapter 5 : Problem Identification & Objective Formulation
Chapter 6 : Design and Development
Chapter 7 : Demonstration and Evaluation
Chapter 8 : Conclusion

CHAPTER SIX: ARTEFACT DESIGN AND DEVELOPMENT

6.1 INTRODUCTION

The primary objective of this study was to develop an artefact which analyses document data stores in terms of structured data model constructs. In order to achieve this, the study formulated a design science research (DSR) methodology discussed in Chapter 2. This chapter presents the third activity of this study's DSR methodology – design and development.

The objective of this chapter is to demonstrate the design and development of the artefact. This activity implements the suggestions made by the researcher in Chapter 5.

This chapter is divided into the following sections aimed at describing the sub-cycles of the design and development activity process: prescriptive knowledge of concepts used to develop the artefact (Section 6.2), functionality of the artefact (Section 6.3), the design and development activity (Section 6.4); and finally, the conclusion (Section 6.5).

6.2 PRESCRIPTIVE KNOWLEDGE OF CONCEPTS USED TO DEVELOP THE ARTEFACT

Certain concepts need to be discussed before the description of the development of the artefact is presented. These concepts were used as prescriptive knowledge in terms of DSR and include algorithms, pseudocode, continuous design and object-oriented design. Figure 6.1 illustrates where these prescriptive knowledge concepts are allocated within this study's DSR useful knowledge base.

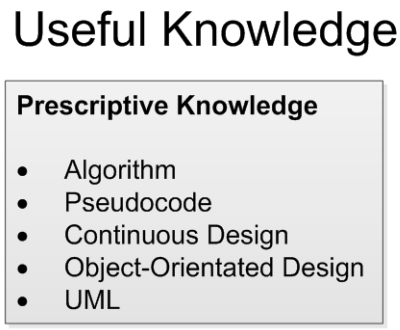


Figure 6.1: DSR useful knowledge base as applicable to this study

6.2.1 Algorithms

An algorithm is defined as any well-defined series of steps that take a set of values, or some value, as input and manipulate it to create a set of values as output (Cormen *et al.*, 2001; Deitel & Deitel, 2006:167; Attaway, 2009:42; Deitel & Deitel, 2013:143). An algorithm can be viewed as a tool for solving well-specified computational problems (Cormen *et al.*, 2001). In modular programming, the problem is broken down into separate steps, and refined until the resulting steps are small enough to handle (Attaway, 2009:42). In short, an algorithm takes input, modifies it and presents output.

In this study, algorithms are developed to analyse NoSQL data and present output. The NoSQL data in BSON format represents the input for the algorithms and the algorithm analyses. The data are then manipulated, resulting in the production of the desired output in the form of a list of possible entities, keys, attributes and relationships.

6.2.2 Pseudocode

Pseudocode helps programmers to develop algorithms in an informal language. The algorithms can be converted to actual programming languages at a later stage. Pseudocode is extremely convenient and user friendly, making it remarkably similar to everyday English, but it is not a real programming language. Pseudocode is not

executed on computers and is seen as a ‘think out’ application used by programmers before they attempt to write it in the actual programming language (Deitel & Deitel, 2006:167; Deitel & Deitel, 2013:144).

In this study, pseudocode is used to illustrate the algorithms developed. Its use is intended to limit the content of the initial programming text and to explain the algorithms in a simplified and understandable language. Pseudocode helps to present the artefact on an abstract level, enabling the artefact to be implemented in various programming environments.

6.2.3 Continuous design

Shore (2004:20) defines continuous design as “the process of using refactoring to continuously improve a program’s design”. Evolutionary design or emergent design are synonyms for continuous design (Shore, 2004:20). When problems arise in continuous design they are solved, or when a new feature does not fit the design, the design is updated. Intensive and constant review of the design is required. In continuous design it is very important to simplify the design (Shore, 2004:22). The simplification of the design bypasses existing code encountered which was developed by someone else, and thereby the need for fixing someone else’s problems, which may be costly, is avoided (Shore, 2004:22). Table 6.1 outlines specific design goals required by continuous design.

Table 6.1: Design goals required for continuous design quoted from Shore (2004:20)

GOAL	DESCRIPTION
Don’t Repeat Yourself (DRY)	There’s little duplication.
Explicit	Code clearly states its purpose, usually without needing comments.
Simple	Specific approaches are preferred over generic ones. Design patterns support features, not extensibility.
Cohesive	Related code and concepts are grouped together.

Decoupled	Unrelated code and concepts can be changed independently.
Isolated	Third-party code is isolated behind a single interface.
Present-day	The design does not try to predict future features.
No hooks	Interfaces, factory methods, events and other extensibility 'hooks' are left out, unless they meet a current need.

In this study continuous design is used to develop the algorithms. The first feature is implemented and then improved on and expanded to accommodate the rest of the features.

6.2.4 Object-oriented design

Object-oriented design (OOD) is described by Bentley and Whitten (2007:648) as "an approach used to specify the software solution in terms of collaborating objects, their attributes, and their methods", and the goal of OOD "is to specify the objects and messages of the system". OOD uses Unified Modelling Language (UML) class diagrams to demonstrate classes and the relationships found between classes. There are various class types available in OOD. They are summarised in Table 6.2.

Table 6.2: OOD class types quoted from Bentley and Whitten (2007:648)

CLASS	DESCRIPTION
Entity	An object class that contains business-related information and implements analysis classes.
Interface	An object class that provides the means by which an actor can interact with the system.
Control	An object class that contains application logic.
Persistence	An object class that provides functionality to read and write in a database.
System	An object class that handles operating system-specific functionality.

In this study OOD is used to assist in the design of the algorithm. OOD class diagrams using UML are used to demonstrate the relationships between these classes. The main class of the algorithm is a control class that implements logic for solving the problem.

The next section introduces the functionality of the artefact.

6.3 FUNCTIONALITY OF THE ARTEFACT

This section demonstrates the overall functionality of the artefact by:

- discussing the suggestions made by the researcher to assist in artefact development;
- demonstrating the suggestions by using an example;
- describing the processing logic of the artefact in the form of an activity diagram; and
- describing how to determine elements and types in document stores.

6.3.1 Suggestions

In Chapter 5 the researcher made suggestions to assist in the development of the artefact. Table 6.3 lists these suggestions.

Table 6.3: Suggestions made by the researcher

#	SUGGESTIONS
1	The initial document is identified as the first entity and possibly also the associative entity.
2	The keys are identified as attributes.
3	Sub-documents and arrays within the document are considered as entities with attributes.
4	The attributes and their values are evaluated for uniqueness in order to identify

	possible primary keys.
5	Sub-documents and arrays within the parent document, already suggested as entities with attributes, resemble one-to-many relationships.

Each of these suggestions is subsequently demonstrated by analysis of example data. Their implementation is demonstrated in the design and development section of this chapter.

6.3.2 Example analysis of a document

The suggestions listed in Section 6.3.1 assist in the artefact development. Figure 6.2 illustrates how the suggestions made by the researcher are applied to the example data used in Chapters 3 and 4.

6.3.3 Activity diagram

This section describes the algorithm logic by using an activity diagram. Figure 6.3 demonstrates the logic used for analysing the documents. It begins at the START node, from there it loops through all the documents one by one. Documents (discussed in Chapter 4) encapsulate value pair formats such as JSON and carry a key identifier to appropriately identify the document. MongoDB contains a key identifier for each document and the documents are stored in BSON, a binary format of JSON. When there are no documents left, the algorithm exits at the END node. The first node after the loop is the document node. This node takes a document and presents its elements. The loop continues to assess each element to check the element types, namely: document (IsTypeDocument), array (IsTypeArray) or other (IsTypeOther). Logic pertaining to type identification is described in Section 6.3.4.

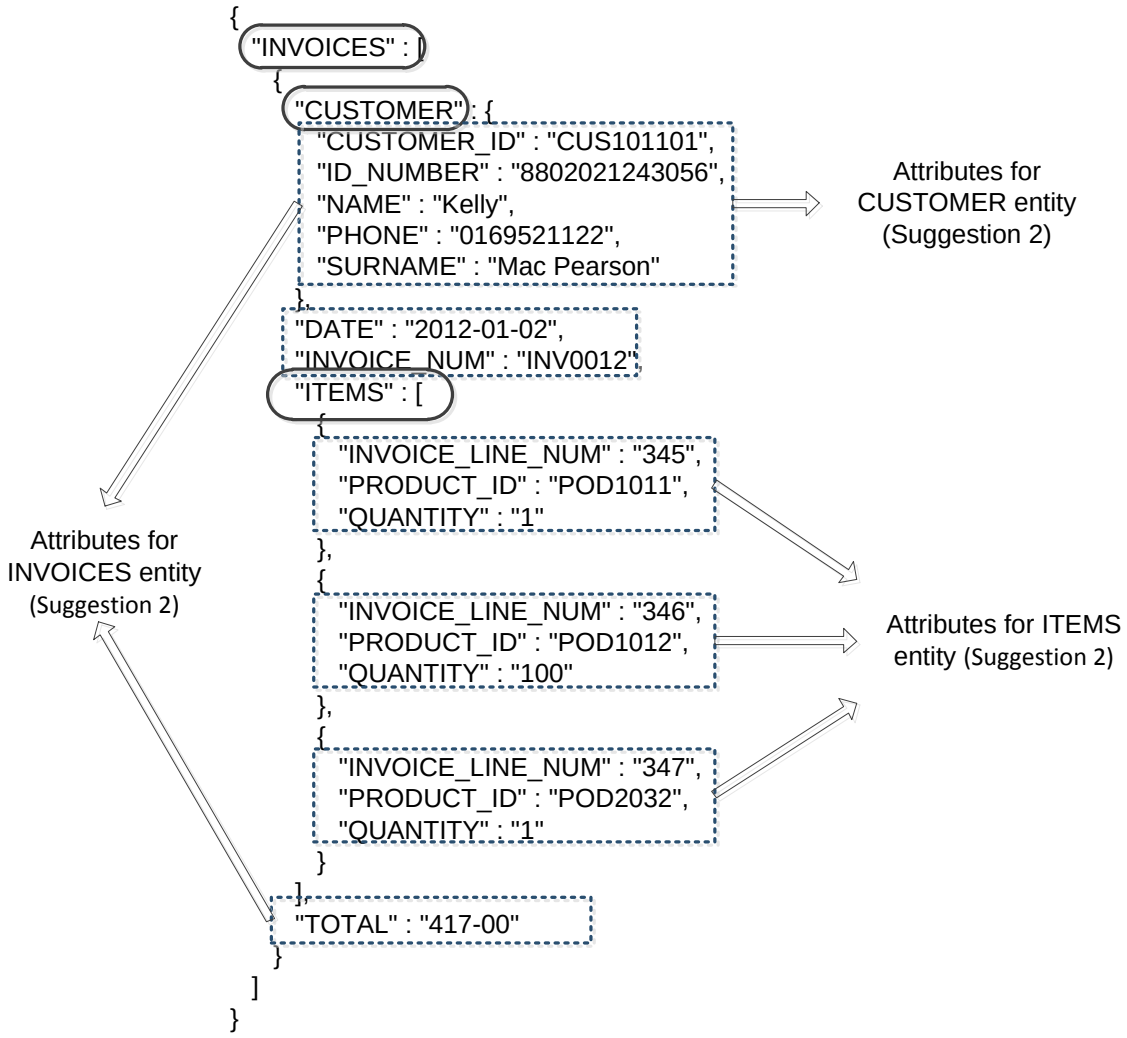


Figure 6.2: Suggestions illustration

If the element is of type document (IsTypeDocument) representing a sub-document in the document, the document becomes an entity (Add_Entity) and a relationship between the document and the element (Add_Relationship) is identified. The algorithm returns to the document node to process the sub-document before continuing with the elements to follow. This is known as recursive programming.

If the element is of type array (IsTypeArray) representing an array list in the document, the document becomes an entity (Add_Entity) and a relationship between the document and the array element (Add_Relationship) is identified. In the array node, each value is looped through. The loop continues to assess each value in

terms of element types: document (IsTypeDocument), array (IsTypeArray) or other (IsTypeOther).

Finally, if the element or value type is of type other (IsTypeOther), the entity is modified, adding the element as an attribute. The algorithm then loops to the next document. This is done in both the functions of the document and the array.

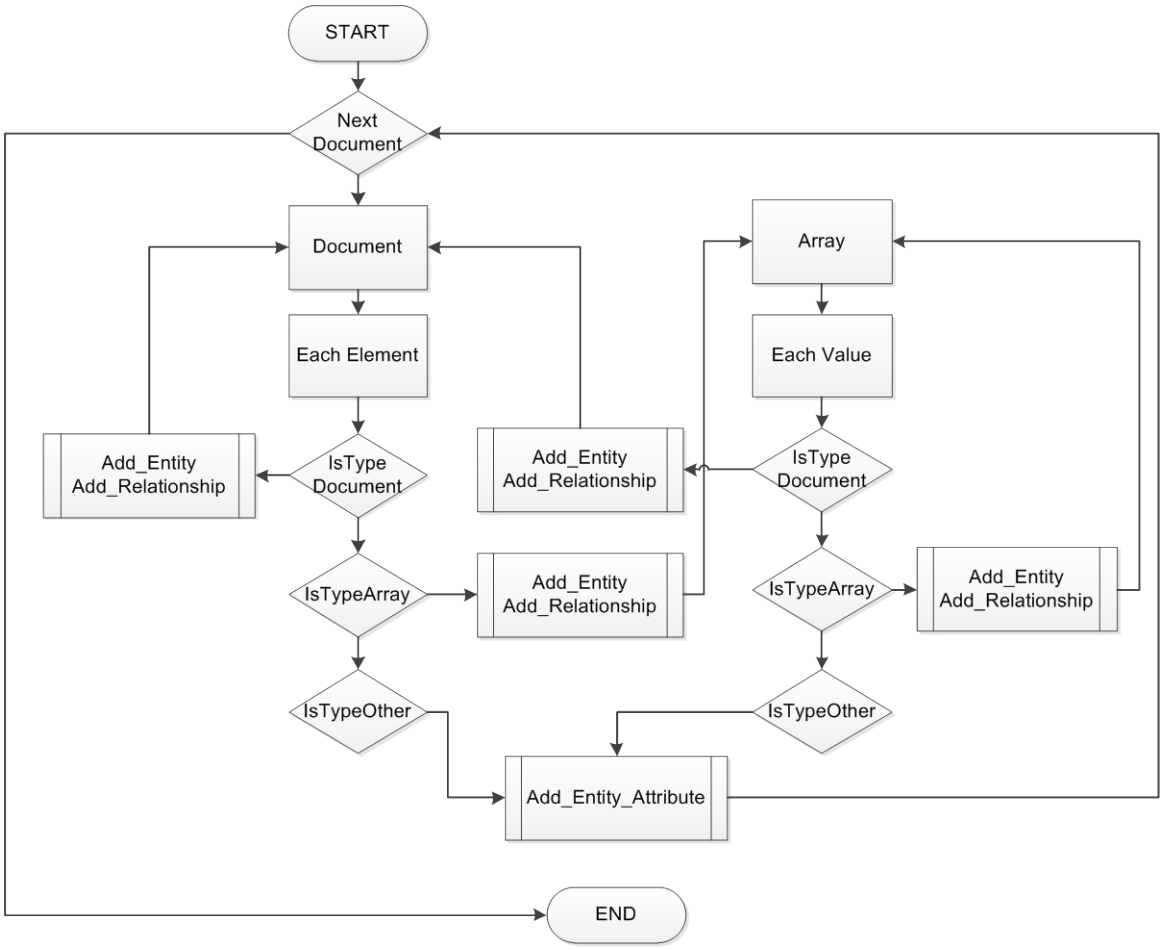


Figure 6.3: Illustration of the document and array processing logic

6.3.4 Determine element and types in document stores

The JSON structure discussed in Chapter 4, Section 4.3.1, uses various groupings of characters to distinguish between certain types of elements. These groupings are important when differentiating between elements. They are also used to determine

the type of element – whether an element of a document is an object or a list. Table 6.4 below illustrates the important characters of JSON type definitions.

Table 6.4: Important formatting characters of JSON

CHARACTER	DESCRIPTION
{	Indication of the start of a document or object
}	Indication of the end of a document or object
[	Indication of the start of a list
]	Indication of the end of a list
,	Element separator
:	Key and value separator

There are numerous coding possibilities to determine the type of element. The following explanation is of the most basic process for determining the element type.

To determine the type of element, one can loop through the characters in the document one-by-one. This should be done when the data are read from the file. Each character is compared to the characters presented in Table 6.4. If a character matches a character in the table, for example the opening curly bracket ({), then it should be indicated as the start of a document. This implies that everything contained within the curly bracket ({ }) is a document. Characters between brackets can be stored in a dictionary list. In the dictionary list, the element is then stored in combination with its type. The dictionary list can be used later to differentiate between the elements and types in the document.

The functions of *IsDocumentType* and *IsTypeArray* shown in Figure 6.3 can then use the dictionary list to determine the type of element. The MongoDB C# driver provides a similar implementation of the process to determine the element type. MongoDB

reads the BSON buffer and loops through each of the characters to determine elements and their types. Elements and their type are stored in a dictionary list and then later used by the MongoDB driver to determine what type and element they are.

6.4 DESIGN AND DEVELOPMENT OF THE ARTEFACT

This section of the chapter describes the process followed for the artefact development and explains the designed algorithms.

This section is divided into two parts: it explains the setup of the initial artefact constructs, which is then followed by an explanation of the cycles of development to modify the artefact. Figure 6.4 demonstrates these cycles, each with their problem, suggestion and developed coding. Each cycle leads toward the next cycle to complete the artefact.

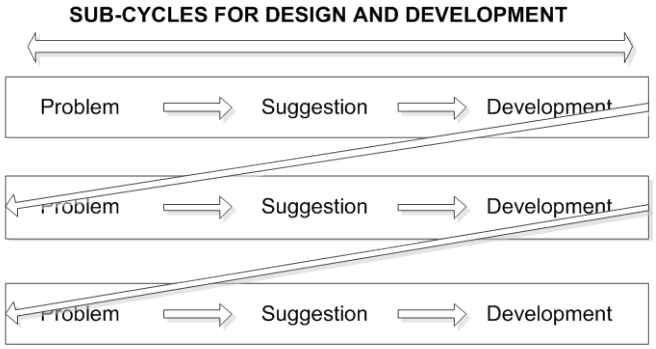


Figure 6.4: Inner design cycles of the design and development activity

The code presented in this section is a variation of pseudocode with some object-oriented design concepts included. The complete pseudocode is given in Appendix C, and the actual implementation code is given in Appendix D. The code presented in Appendix D is written in VB.NET. Throughout the development, code segments are given to illustrate the algorithms and, where possible, a graphical representation is given.

MongoDB contains a set of drivers which easily interacts with the MongoDB database on a programming level as discussed in Chapter 4. The drivers include some of the following source languages: C#, Java, PHP, Phython and Ruby (MongoDB, 2007). The C# driver was used in the VB.NET application because these two languages share a common library runtime. The source code of the MongoDB C# driver is extensive and is omitted in this study. However, the source code is indicated where certain functions or methods of the driver are used.

Code segments are referred to using the notation (CS1.1:1), meaning code chapter 1, code segment 1, line 1. The names of variables, classes, functions and lists in the text are shown in italics to distinguish them from explanation text. Table 6.5 lists the adjectives used before variables, classes, functions and lists to clearly distinguish between the different types of coding and text.

Table 6.5: Pre-text used to distinguish between the different types

ADJECTIVE	TYPE	EXAMPLE
var_	Instance variable	var_name, var_count
lst_	List	lst_Entities, lst_Attributes
obj_	Object	obj_currentEntity, obj_newEntity
cls_	Class	cls_Entity, cls_Attribute

This study specifically implements the List class as set of default functions and methods available. These functions and methods are described in Table 6.6.

Table 6.6: List of default implemented functions and methods

FUNCTION/METHOD	DESCRIPTION
add(object)	This function adds an object to the list.

removeAt(i)	This function removes an object from the specified index.
clear()	This method clears all objects within the list.
contains(string_literal)	This function evaluates whether the list already contains the specified element.
containsKey(string_literal)	This function evaluates whether the list already contains the specified key.
containsValue(object_literal)	This function evaluates whether the list already contains the specified value.
item(i or string_literal) **OR** **listname(i or string_literal)**	Returns the specific object from the list to the specified index or string literal reference. The list name refers to the list object's name (example: lst_Attributes)

6.4.1 Setup of variables, lists and classes

This section provides an explanation of the initial entity classes used by the artefact. The classes are illustrated, explained and the initial construct of the *cls_Document_store_analyser* class is discussed.

The *cls_Document_store_analyser* class is a complete library of all the algorithms, providing a control class for the algorithm logic.

The entity classes *cls_Entity, cls_Attribute,* and *cls_Value* are related. The *cls_Entity* class uses the *cls_Attribute* class and the *cls_Attribute* class uses the *cls_Value* class. The control class *cls_Document_store_analyser* uses the *cls_Entity* class.

Code segment 6.1 is an illustration of the *cls_Entity* class used in the artefact. The function of this class is to store the *var_name* and its possible *lst_Attributes* in a list type for the entity. Figure 6.5 shows the UML class diagram for the *cls_Entity* class.

```
1   class cls_Entity
2           String var_name
3           List lst_Attributes
4   end class
```

Code segment 6.1: Class construct: cls_Entity.

Code segment 6.1, line 3, is a list, because an entity may contain a list of attributes. In UML, the minus (-) sign indicates that the variable or object or method is private to that class, and the plus (+) sign indicates that they are public. For the purpose of this study, all methods, functions and variables are declared public. This eliminates the need to create mutator and accessor methods, which may clutter the coding and representation of the algorithm of the artefact. To eliminate code cluttering, the actual code implementation, available in Appendix D, does not implement mutator and accessor methods for classes.

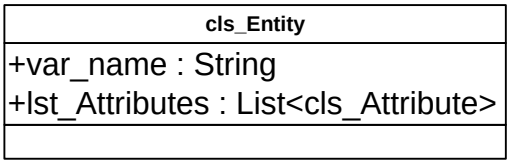


Figure 6.5: UML class diagram: cls_Entity

Code segment 6.2 is an illustration of the *cls_Attribute* class that is used in the entity class. The function of this class is to store the *var_name* of the attribute and the list of *lst_Values* found for that attribute. Figure 6.6 shows the UML class diagram for the *cls_Attribute* class.

```
1    class cls_Attribute
2        String var_name
3        List lst_Values
4    end class
```

Code segment 6.2: Class construct: cls_Attribute

Code segment 6.2, line 3, is a list, because an attribute may contain many values.

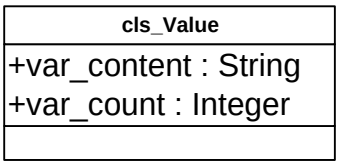


Figure 6.6: UML class diagram: cls_Attribute

Code segment 6.3 is an illustration of the *cls_Value* class that is used in the *cls_Attribute* class above. The function of this class is to store the value of the attribute in *var_content* and *var_count* representing the number of times the value occurs. This count helps to identify possible primary keys at a later stage. Figure 6.7 is the UML class diagram for the *cls_Value* class. Figure 6.8 is the UML class diagram for *cls_Entity*, *cls_Attribute* and *cls_Value* classes showing the relationships among them.

```
1    class cls_Value
2        String var_content
3        Integer var_count
4    end class
```

Code segment 6.3: Class construct: cls_Value

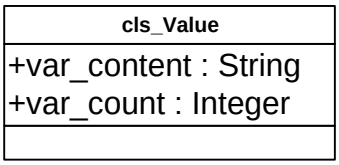


Figure 6.7: UML class diagram: cls_Value

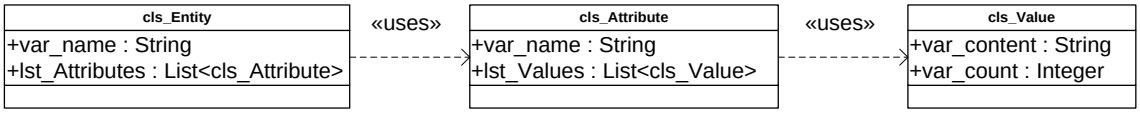


Figure 6.8: Relationships between cls_Entity, cls_Attribute and cls_Value

Code segment 6.4 is an illustration of the *cls_relationship* class that is used to store all possible relationships found. The string *var_Entity_one* stores the parent entity name and the string *var_Entity_many* stores a related child entity. Figure 6.9 shows the UML class diagram for the relationship class.

```
1   class cls_Relationship
2           String var_Entity_one
3           String var_Entity_many
4   end class
```

Code segment 6.4: Class construct: cls_Relationship

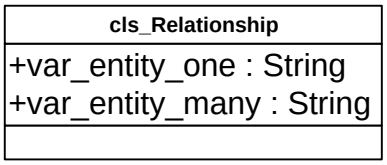


Figure 6.9: UML class diagram: cls_Relationship

Code segment 6.5 is the initial construct of the *cls_Document_store_analyser* class. This class is the control class used in this artefact. All algorithms are added to this class and at the end of the development cycle the complete class is shown with all modifications in Appendix C.

In the *cls_Document_store_analyser* class, lists are created for entities and relationships (CS6.5:2-3).

The function *add_entity* (CS6.5:5-14) adds an identified entity to the entity list (CS6.5:2). It checks the entity list (CS6.5:6) to see if the entity has already been added. If it was not added a new entity object is created from the entity class and then added to the list.

The function *add_entity_attribute* (CS6.5:16-23) updates an existing entity to add an attribute to that entity. The function checks if the attribute has been added to the entity in the entity list (CS6.5:2) and if it has not been added, it will be added to the attribute list of the entity.

The function *add_relationship* (CS6.5:29-38) adds an identified relationship between two entities to the relationship list (CS6.5:3). It checks whether the relationship already exists between the two entities. If it does not exist, it creates a relationship object (CS6.5:33-36) using the *cls_relationship* class and adds it to the list. This function is used in the third cycle of the development phase.

Figure 6.10 shows the UML class diagram for the *cls_Document_store_analyser* class and the relationships with other classes. In Figure 6.10, it is seen that the *cls_Document_store_analyser* class implements methods such as *add_entity*, *add_entity_attribute* and *add_relationship*.

```
1   class cls_Document_store_analyser
2           List lst_Entities
3           List lst_Relationships
4
5           function add_entity(name)
6               if lst_Entities.contains(name)
7                   // do nothing
8               else
9                   cls_Entity obj_newEntity  = new cls_Entity()
10                  obj_newEntity.var_name  = name
11                  obj_newEntity.lst_Attributes  = new List
12                  lst_Entities.add(obj_newEntity)
13              end if
14          end function
15
16          function add_entity_attribute(name,  attribute)
17              cls_Entity obj_currentEntity  = lst_Entities(name)
18              if obj_currentEntity.lst_Attributes.contains(attribute)
19                  // do nothing
20              else
21                  cls_Attribute obj_newAttribute   = new cls_Attribute()
22                  obj_newAttribute.var_name  = attribute
23                  cls_Values  obj_newValue  = new cls_Value()
24                  obj_newAttribute.lst_Value  = obj_newValue
25                  obj_currentEntity.lst_Attributes.add(obj_newAttribute)
26              end if
27          end function
28
29          function add_relationship(entity_one,  entity_many)
30              if lst_Relationships.contains(entity_one  + entity_many)
31                  // do nothing
32              else
33                  cls_Relationship  obj_newRelationship  = new cls_Relationship()
34                  obj_newRelationship.var_Entity_one  = entity_one
35                  obj_newRelationship.var_Entity_many  = entity_many
36                  lst_Relationships.add(obj_newRelationship)
37              end if
38          end function
39  end class
```

Code segment 6.5: Initial construct of the cls_Document_store_analyser class

These methods accept input parameters that are used in these methods. The input parameters are preceded by “in”, to indicate a receiving parameter as opposed to “out”, which is a return parameter.

The initial construct of the *cls_Document_store_analyser* class has been discussed. The development cycles, as illustrated in Figure 6.4, are explained in Section 4.2.2.

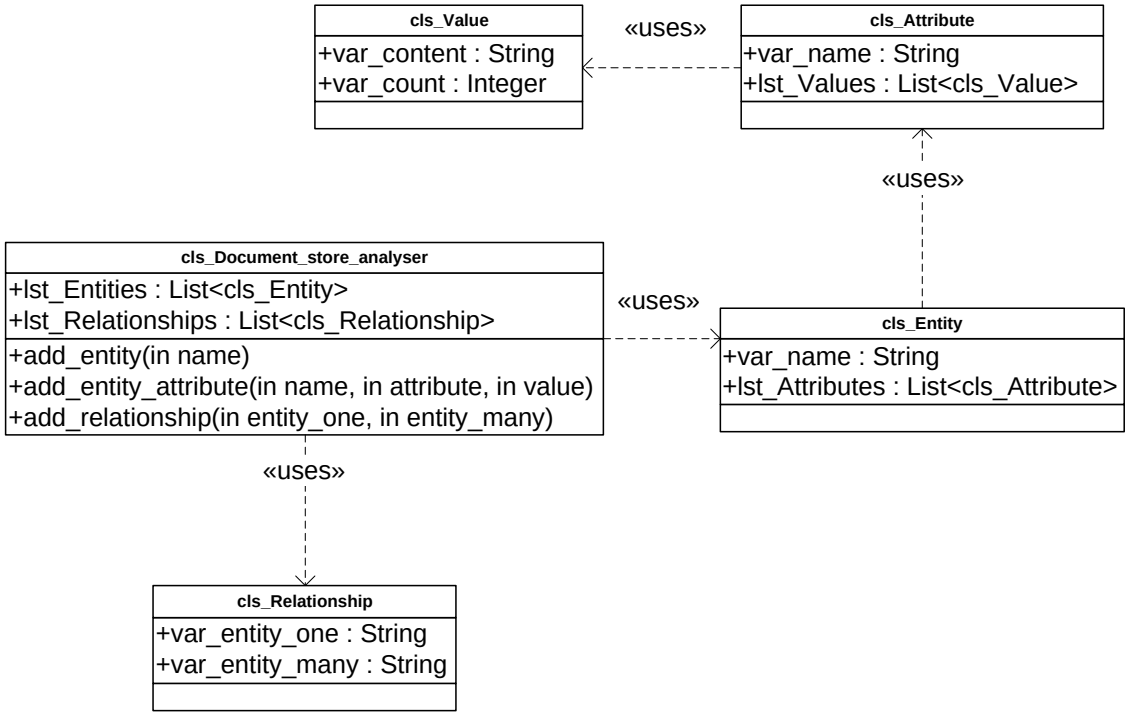


Figure 6.10: UML class diagram: cls_Document_store_analyser (first iteration)

6.4.2 Cycle 1: Entity and attribute identification

Cycle 1 addresses the first, second and third design activities suggested in Table 6.3. These suggestions are that:

1. the initial document should be identified as the first entity and possibly also the associative entity;
2. the keys should be identified as attributes; and
3. sub-documents and arrays within the document should be considered as entities with attributes.

6.4.2.1 Problem and suggestion

The researcher suggests that the initial document should become the first entity for the analysis. The elements that are of type document or type array are considered to be entities and their elements are considered to be their attributes. All elements in this document that are not of type document or type array are considered to be attributes. A recursive algorithm is suggested for the processing of inner documents until no more are found. These entities and attributes are stored in an array list to be presented as output and interpreted by the user during the evaluation phase (described in Chapter 7).

6.4.2.2 Development

Code segment 6.6 expands the *cls_Document_store_analyser* class and illustrates the solution to identifying possible entities and attributes.

Code segment 6.5 given in Section 6.4.1 illustrates the identification of entities and attributes to the entity list (refer CS6.5:2). The first routine of CS6.6, namely *initiation*, is the starting point for the algorithm (CS6.6:1). The first few lines are comments indicating the connection to the server and obtaining collections for a database. Most of the MongoDB driver is used here to connect to the server and obtain the database, collections and documents.

The for-each statement is the actual start of the algorithm (CS6.6:6-8). Within this for-each statement loop, the documents retrieved in the collection are passed to the document function (CS6.6:11) for processing. The *document_analysis* function (CS6.6:11-24) accepts the parameters of a document and the name of the parent entity. In the first instance, the name of the collection is used as the parent entity.

```
routine initiation()
    //connect to server
    //get List of collections.
    //loop thru collections
    //loop thru documents  in collection
    for each doc in collection
        document_analysis(doc,  collection.name)
    loop
end routine

function document_analysis(doc,  parent)
    List lst_Elements = doc.elements
    for each element  in lst_Elements
        if element.istypeDocument
            add_entity(element.name)
            document_analysis(element.value,  element.name)
        elseif element.istypeArray
            add_entity(element.name)
            array_analysis(element.value,  element.name)
        else
            add_entity_attribute(parent,  element.name)
        end if
    loop
end function

function array_analysis(arr,  parent)
    List lst_Values = arr.values
    for each value in lst_Values
        if element.istypeDocument
            add_entity(values.name)
            document_analysis(values.value,  values.name)
        elseif element.istypeArray
            add_entity(values.name)
            array_analysis(values.value,  values.name)
        else
            add_entity_attribute(parent,  values.name)
        end if
    loop
end function
```

Code segment 6.6: Adding entities and attributes in the cls_Document_store_analyser class

In the *document_analysis* function, the function starts by retrieving a list of all elements from the *doc* received parameter and stores it in an elements list (CS6.6:12). It continues to loop through all elements with a for-each statement (CS6.6:13-23). Since a document can contain sub-documents and arrays (discussed in Chapter 4, Section 4.3.3), an if-statement is used to check the type of element (CS6.6:14-22). If the element is of type document or type array, then it is accepted to be a potential entity and it is added to the entity list (CS6.6:15 and 18). The checking of the type of element is a function available in the MongoDB driver. If the element is a document, then it is again passed to the *document_analysis* function (CS6.6:16) for further processing. This forms a recursive loop. The recursive loop enables the algorithm to work through the entire document.

The same principle is applied when the element is of type array (CS6.6:17). The *array_analysis* function is called (CS6.6:19) for further processing of array type values. The *array_analysis* function (CS6.6:26-38) is similar to that of the document function, with the only difference being that a list of values instead of a list of elements (CS6.6:27) is retrieved. The function then continues to loop through all values with a for-each statement (CS6.6:28-38). The values found in the array could also be documents and arrays (discussed in Chapter 4, Section 4.3.3); therefore, the process of checking the value is presented again. Should the type be of type document or array then the function is called for processing once more. Both functions, *document_analysis* and *array_analysis*, form a recursive loop. The else statements found in both functions (CS6.6:20 and 35) accept that when no more documents or arrays exist, that element or value is an attribute. The *add_entity_attribute* function is then called to add the attribute to the entity (refer CS6.5:16). In this way, the entity list presented in CS6.5 (refer CS6.5:2) is generated with possible entities and their attributes. These functions solve the problem of the entity and attribute identification.

Figure 6.11 shows the final UML class diagram for the *cls_Document_store_analyser* class including all methods. In Figure 6.11 it is seen that the *cls_Document_store_analyser* class implements the initial methods

add_entity, *add_entity_attribute* and *add_relationship*, after which the methods, *initiation*, *document_analysis* and *array_analysis* are added.

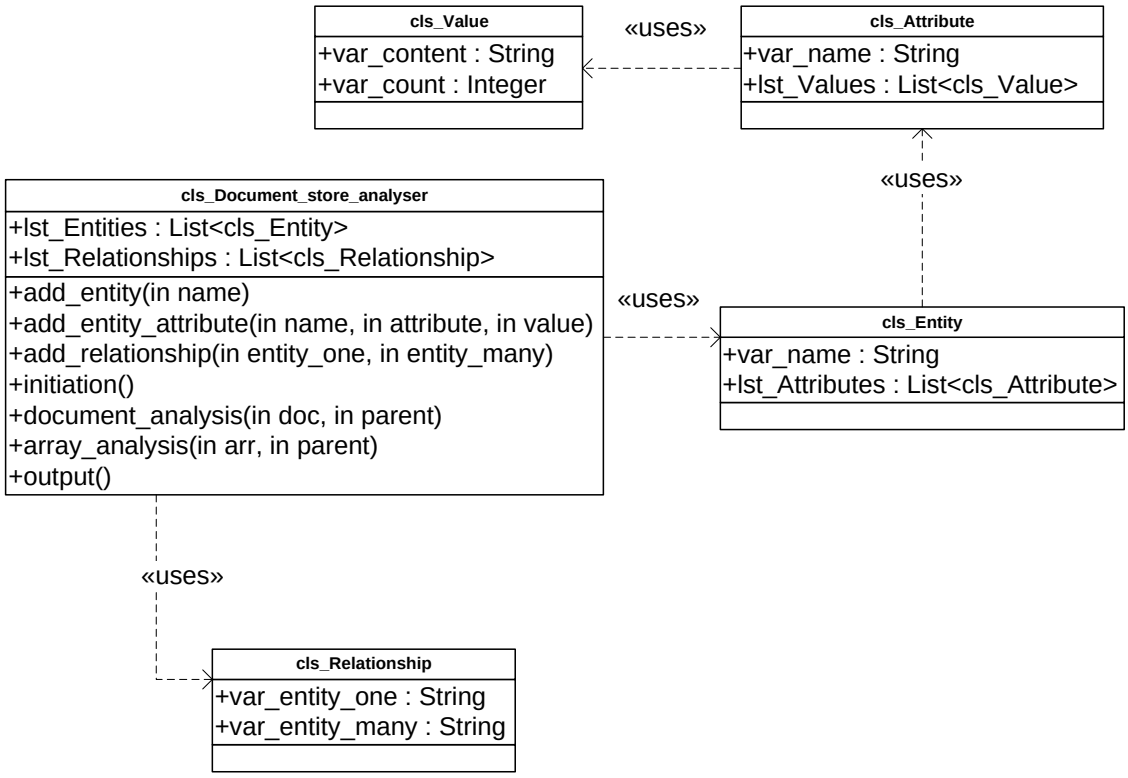


Figure 6.11: UML class diagram: cls_Document_store_analyser (final iteration)

6.4.3 Cycle 2: Primary key identification

Cycle 2 addresses the fourth design suggestion made by the researcher, which states that the attributes and their values are evaluated for uniqueness in order to identify possible primary keys.

6.4.3.1 Problem and suggestion

The researcher suggests that a count should be maintained for each attribute that is identified in the document, sub-documents or arrays. The value for each attribute, as well as the number of occurrences, is stored in an array. The uniqueness of the attribute can be tested by accepting that if the count of all the values is not more than one, the value is unique and may be identified as a primary key. These counts of

values for attributes are stored in an array list to be presented as output and interpreted by the user during the evaluation phase.

6.4.3.2 Development

Code segment 6.7 illustrates the solution to identifying possible primary keys. This code segment is a modified version of the function *add_entity_attribute* (refer CS6.5:16). The code is explained as follows:

```
1   function add_entity_attribute(name,  attribute, value)
2       cls_Entity obj_currentEntity = lst_Entities(name)
3       if obj_currentEntity.lst_Attributes.contains(attribute)
4           cls_Attribute obj_currentAttribute = obj_currentEntity.lst_Attributes(attribute)
5           if currentAttribute.values.contains(value)
6               obj_currentAttribute.lst_Values(value).var_count =
7                   obj_currentAttribute.lst_Values(value).var_count + 1
8               obj_currentEntity.lst_Attributes(attribute) = obj_currentAttribute
9           else
10              cls_Value obj_currentValue = new cls_Value()
11              obj_currentValue.value = value
12              obj_currentEntity.lst_Attributes(attribute).lst_Values.add(obj_currentValue)
13          end if
14      else
15          cls_Attribute obj_newAttribute = new cls_Attribute()
16          obj_newAttribute.var_name = attribute
17          cls_Value obj_newValues = new cls_Value()
18          obj_newValues.var_content = value
19          obj_newValues.var_count = 1
20          obj_newAttribute.lst_Values = obj_newValues
21          obj_currentEntity.lst_Attributes.add(obj_newAttribute)
22      end if
23  end function
```

Code segment 6.7: Modified add_entity_attribute function for finding primary keys

Code segment 6.7 illustrates a modified *add_entity_attribute* function (CS6.7:1-23), which is the first step in identifying possible primary keys. The existing entity is

retrieved from the entities list using the supplied parameter “name”. This enables the algorithm to appropriately update the correct entity. Possible primary keys are identified by capturing the values for attributes identified and keeping a record of the number of times the attribute occurred.

It can be argued that if the count of a value for an attribute is greater than 1, then that attribute cannot be a possible primary key. In the function *add_entity_attribute* (CS6.7:1-23), a new parameter is added, namely “value” (CS6.7:1). If the attribute does not exist in the entity, it is added to the entity with the current value (CS6.7:14-22). If the attribute exists, the value is added to the values list of the attribute (CS6.7:-13) or the count for that value is updated (CS6.7:5-9). This modified function solves the problem of identifying possible primary keys.

6.4.4 Cycle 3: Relationship identification

Cycle 3 addresses the fifth design suggestion made by the researcher. The fifth suggestion dictates that sub-documents and arrays within the parent document, already suggested as an entity with attributes, resemble one-to-many relationships.

6.4.4.1 Problem and suggestion

The researcher suggests that there is a one-to-one or one-to-many relationship between a document and a sub-document, and an array or sub-array. These relationships are stored in an array to be presented as output and interpreted by the user during the evaluation phase.

6.4.4.2 Development

Code segment 6.8 below illustrates the solution for identifying possible relationships between the entities. This code segment is a modified version of the document function and the array function (refer CS6.6:11 and 26).

Code segment 6.8 is a modified version of the document and array functions, which includes the addition of possible relationships given in code segment 6. This is done

by adding the *add_relationship* function (CS6.8:16, 20, 33, 37) and passing the current parent value and the current element name. The function described here adds this relation to the relationship list (refer CS6.5:2). These relationships will be presented to the user for interpretation. This modification and addition solves the problem of identifying possible relationships.

A limitation of the artefact is that it cannot identify the types of relationships between entities, such as one-to-one and one-to-many. A one-to-many relationship is the most commonly found relationship between entities; this artefact uses the *var_Entity_many* variable to illustrate the child entity. The relationships are listed for the user to be interpreted. Many-to-many relationships are not possible for the artefact to identify. Only by interpreting the result and possibly combining one-to-many relationships, may many-to-many relationships be identified by the user. This limitation of not identifying the type of relationships presents opportunities for future research.

```
routine initiation() // modified
    //connect to server
    //get List of collections.
    //loop thru collections
    //loop thru documents in collection
    for each doc in collection
        document_analysis(doc, collection.name)
    loop
end routine

function document_analysis(doc, parent)
    List lst_Elements = doc.elements
    for each element in lst_Elements
        if element.istypeDocument
            add_entity(element.name)
            add_relationship(parent, element.name)
            document_analysis(element.value, element.name)
        elseif element.istypeArray
            add_entity(element.name)
            add_relationship(parent, values.name)
            array_analysis(element.value, element.name)
        else
            add_entity_attribute(parent, element.name)
        end if
    loop
end function

function array_analysis(arr, parent)
    List lst_Values = arr.values
    for each value in lst_Values
        if element.istypeDocument
            add_entity(values.name)
            add_relationship(parent, element.name)
            document_analysis(values.value, values.name)
        elseif element.istypeArray
            add_entity(values.name)
            add_relationship(parent, values.name)
```

```
38                          array_analysis(values.value,  values.name)
39                      else
40                          add_entity_attribute(parent,  values.name)
41                      end if
42                  loop
43              end function
```

Code segment 6.8: Modified document_analysis and array_analysis for adding relationships

6.4.5 Cycle 4: Generating output

Cycle 4 does not address any of the problems or suggestions made by the researcher. However, this cycle is used to generate output based on all the data stored in the *lst_Entities, lst_Attributes, lst_Value* and *lst_Relationships* lists.

Code segment 6.9 illustrates the *output* function (CS6.9:1-34). The function starts by declaring a string variable *var_output*, to which all other output is concatenated (CS6.9:2). The output string initiation is followed with a for-each loop that loops through all the entities in the *lst_Entities* list (CS6.9:4-25). The *var_output* variable is updated by concatenating the entity name to the variable (CS6.9:5). This is followed by an if-statement to check whether that entity does contain attributes (CS6.9:5). If the entity contains attributes, another for-each loop is used to loop through the *lst_Attributes* list (CS6.9:8-20). A *var_value_count* variable is initiated to be used later to differentiate between primary keys and non-primary key attributes. The algorithm loops through each value of the *lst_Values* list and updates the *var_value_count* value if the count is greater than one (CS6.9:10-14). Should this be the case, then it is not a primary key because the value is repeated multiple times (CS9:16). The *var_output* variable is concatenated with the attribute name. If the *var_value_count* is not greater than one, then it is a primary key and the *var_output* variable is concatenated with the attribute name and a “(PK)” indicator (CS6.9:18). Finally, the *var_output* is concatenated with a comma to differentiate between attributes (CS6.9:22).

```
1   function output()
2       String var_output = "Entities and Attributes:" + NewLine
3
4           for each entity in lst_Entities
5               var_output = var_output + entity.name + "("
6
7               if entity.Attribute.Count > 0
8                   for each attribute in entitly.lst_Attributes
9                       Integer var_value_count = 0
10                      for each value in attribute.lst_Values
11                          if var_value_count < value.var_count
12                              var_value_count = value.var_count
13                          end if
14                      loop
15                      if var_value_count > 1
16                          var_output = var_output + attribute.var_name + ","
17                      else
18                          var_output = var_output + attribute.var_name + " (PK) ,"
19                      end if
20                  loop
21              else
22                  var_output = var_output + ","
23              end if
24                  //remove last 2 characters with subString from output and add NewLine
25          loop
26
27      var_output = "Relationships:" + NewLine
28      for each relationship in lst_Relationships
29          var_output = var_output + relationship.var_Entity_one + "---------<" +
30              relationship.var_Entity_many + NewLine
31      loop
32
33      return var_output
34
35  end function
```

Code segment 6.9: Output function to generate output

The algorithm continues to generate the output for relationships between entities. The *var_output* variable is concatenated with relationships (CS6.9:27) followed by a for-each loop that loops thought all the relationships in the *lst_Relationships* list (CS6.9:28-31). The *var_output* variable is concatenated with the first relationship entity name *var_Entity_one*, linked with "--------<" to indicate a one-to-many relationship, and finally concatenated by the second relationship entity name *var_Entity_many* (CS6.9:29-30).

```
Entities and Attributes:
( _id (pk) )
customer ( customer_id (pk) , id_number (pk) , name (pk) , phone (pk) , surname (pk) )
invoices ( date (pk) , invoice_num (pk) , total (pk) )
items ( invoice_line_num (pk) , product_id (pk) , quantity )

Relationships:
-------<invoices
invoices-------<customer
invoices-------<items
```

Figure 6.12: Generated output of Figure 6.1

The output that is generated with the *output* function of the algorithm is used in Chapter 7 to determine the acceptability and utility of the artefact.

6.5 CHAPTER CONCLUSION

The objective of this chapter was to demonstrate the design and development of the artefact in order to address the main objective of this study. This objective was achieved through the actual development of the artefact using the design cycles suggested by Vaishnavi and Kuechler (2004). The code in this chapter is presented

using pseudocode (given in Appendix C). The actual code implementation of the artefact for testing and evaluation purposes can be found in Appendix D.

Table 6.7 is a summary of each suggestion made throughout the design process as well as the cycles and sections they were addressed in.

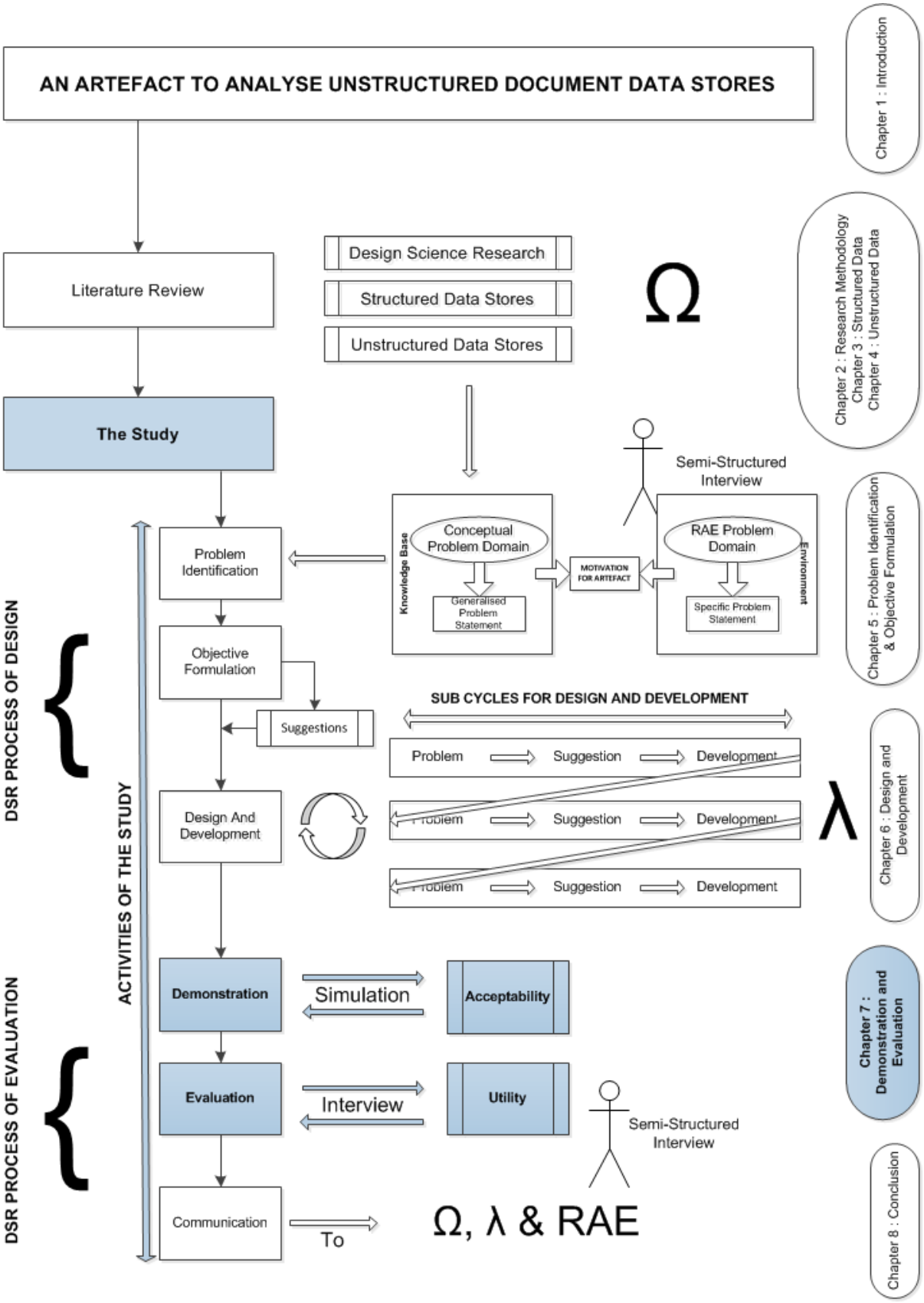
Table 6.7: Suggestions and cycles within addressed sections

#	SUGGESTIONS	ADDRESSED IN
1	The initial document is identified as the first entity and possibly also the associative entity.	Cycle 1, Section 6.4.2.
2	The keys are identified as attributes.	Cycle 1, Section 6.4.2. and Cycle 4, Section 6.4.5.
3	Sub-documents and arrays within the document are considered as entities with attributes.	Cycle 1, Section 6.4.2.
4	The attributes and their values are evaluated for uniqueness in order to identify possible primary keys.	Cycle 2, Section 6.4.3.
5	Sub-documents and arrays within the parent document, already suggested as entities with attributes, resemble one-to-many relationships.	Cycle 3, Section 6.4.4.

Internal testing was conducted during the development of the algorithm. This internal testing was not documented, since formal testing of the artefact follows in the next chapter. It should be noted that the artefact does not make provision for faulty data at this stage. Exceptions are partly handled by the MongoDB C# driver and some exception handling was added to the VB.NET implementation, but more exception handling should be added to ensure a robust product.

The next chapter introduces the demonstration and evaluation of the artefact's acceptability and utility.

(PAGE INTENTIONALLY LEFT BLANK)


AN ARTEFACT TO ANALYSE UNSTRUCTURED DOCUMENT DATA STORES
Literature Review
The Study
Design Science Research
Structured Data Stores
Unstructured Data Stores
Ω
Semi-Structured Interview
Problem Identification
Objective Formulation
Suggestions
Design And Development
Demonstration
Evaluation
Communication
To
Knowledge Base
Conceptual Problem Domain
Generalised Problem Statement
MOTIVATION FOR ARTEFACT
RAE Problem Domain
Specific Problem Statement
Environment
SUB CYCLES FOR DESIGN AND DEVELOPMENT
Problem
Suggestion
Development
λ
Simulation
Acceptability
Interview
Utility
Semi-Structured Interview
Ω, λ & RAE
DSR PROCESS OF DESIGN
DSR PROCESS OF EVALUATION
ACTIVITIES OF THE STUDY
Chapter 1 : Introduction
Chapter 2 : Research Methodology
Chapter 3 : Structured Data
Chapter 4 : Unstructured Data
Chapter 5 : Problem Identification & Objective Formulation
Chapter 6 : Design and Development
Chapter 7 : Demonstration and Evaluation
Chapter 8 : Conclusion

CHAPTER SEVEN: ARTEFACT DEMONSTRATION AND EVALUATION

7.1 INTRODUCTION

The primary objective of this study was to develop an artefact which analyses document data stores in terms of structured data model constructs. In order to achieve this, a design science research (DSR) methodology was formulated (Chapter 2). This chapter presents the fourth and fifth activities of this study's DSR methodology – demonstration and evaluation.

Demonstration is the action of showing or illustrating something (Oxford Dictionaries & Press, 2013). In the context of this DSR study, demonstration refers to the provision of evidence in which the solution to more than one instance of the problem domain by the artefact is provided (Peffers *et al.*, 2008:55).

Evaluation is the action of judging materials or methods in terms of their internal accuracy and consistency or by comparing them with external criteria (Saunders *et al.*, 2009:591). In terms of this DSR study, evaluation is a measurement of how well the solution to the RAE problem domain is supported by the artefact (Peffers *et al.*, 2008:56).

The objective of this chapter is to evaluate the artefact in terms of conceptual acceptability and utility in the real application environment (RAE) in order to achieve credibility as a DSR project (Gregor & Hevner, 2013:342).

This DSR study specifically evaluates the artefact using two criteria – acceptability and utility. The evaluation of acceptability using simulated data represents the demonstration activity of this study. The artefact's utility is evaluated by checking whether it solves the real problem within the application area; this forms the

evaluation activity of this DSR study. These two criteria for evaluation are the main issues that are addressed in this chapter and are listed in Table 7.1.

Table 7.1: Main evaluation issues

ISSUE	DESCRIPTION	VALUE	ADDRESSED IN
1	Does the artefact present sufficient and acceptable output based on the simulated data?	This demonstrates the artefact's value in terms of quality.	Section 7.2
2	Is the artefact usable for the RAE?	This evaluates the artefact's value in terms of utility.	Section 7.3

The discussion in this chapter is divided into the following sections: evaluation of acceptability of output (Section 7.2), evaluation of utility (Section 7.3), demonstration and evaluation summary (Section 7.4), design and development cycle beyond the initial DSR study (Section 7.5), and finally, the conclusion (Section 7.6).

7.2 DEMONSTRATION OF ACCEPTABILITY OF OUTPUT

This section demonstrates the artefact's acceptability of output by using simulated data.

White box testing can be described as the process wherein a system is given input in order to analyse the system processes for generating the required output (Khan, 2010:11). White box testing is implemented to some extent in this section of the DSR study. The artefact is tested with data from various sources to see whether all code segments of the artefact are executed correctly.

Evaluation questions are designed for the assessment of the simulated data. Table 7.2 summarises the questions that the artefact needs to address using simulated data. Table 7.2 is a rubric of measurement for the artefact and the simulated data. In

Table 7.2, each question addresses specific code segments of the complete algorithm. The algorithm is divided into code segments, which are listed next.

```
1     function add_entity(name)
2         if lst_Entities.contains(name)
3             // do nothing
4         else
5             cls_Entity obj_newEntity = new cls_Entity()
6             obj_newEntity.var_name = name
7             obj_newEntity.lst_Attributes = new List
8             lst_Entities.add(obj_newEntity)
9         end if
10    end function
```

Code segment 7.1: Function: add_entity

```
1   function add_entity_attribute(name, attribute, value)
2       cls_Entity obj_currentEntity = lst_Entities(name)
3       if obj_currentEntity.lst_Attributes.contains(attribute)
4           cls_Attribute obj_currentAttribute = obj_currentEntity.lst_Attributes(attribute)
5           if currentAttribute.values.contains(value)
6               obj_currentAttribute.lst_Values(value).var_count =
7                   obj_currentAttribute.lst_Values(value).var_count + 1
8               obj_currentEntity.lst_Attributes(attribute) = obj_currentAttribute
9           else
10              cls_Value obj_currentValue = new cls_Value()
11              obj_currentValue.value = value
12              obj_currentEntity.lst_Attributes(attribute).lst_Values.add(obj_currentValue)
13          end if
14      else
15          cls_Attribute obj_newAttribute = new cls_Attribute()
16          obj_newAttribute.var_name = attribute
17          cls_Value obj_newValues = new cls_Value()
18          obj_newValues.var_content = value
19          obj_newValues.var_count = 1
20          obj_newAttribute.lst_Values = obj_newValues
21          obj_currentEntity.lst_Attributes.add(obj_newAttribute)
22      end if
23  end function
```

Code segment 7.2: Function: add_entity_attribute

```
1   function add_relationship(entity_one, entity_many)
2       if lst_Relationships.contains(entity_one + entity_many)
3           // do nothing
4       else
5           cls_Relationship obj_newRelationship = new cls_Relationship()
6           obj_newRelationship.var_Entity_one = entity_one
7           obj_newRelationship.var_Entity_many = entity_many
8           lst_Relationships.add(obj_newRelationship)
9       end if
10  end function
```

Code segment 7.3: Function: add_relationship

```
1   function document_analysis(doc, parent)
2           List lst_Elements = doc.elements
3           for each element in lst_Elements
4                   if element.istypeDocument
5                       add_entity(element.name)
6                       add_relationship(parent, element.name)
7                       document_analysis(element.value, element.name)
8                   elseif element.istypeArray
9                       add_entity(element.name)
10                      add_relationship(parent, values.name)
11                      array_analysis(element.value, element.name)
12                  else
13                      add_entity_attribute(parent, element.name)
14                  end if
15          loop
16  end function
```

Code segment 7.4: Method: document_analysis

```
1   function array_analysis(arr, parent)
2           List lst_Values = arr.values
3           for each value in lst_Values
4                   if element.istypeDocument
5                       add_entity(values.name)
6                       add_relationship(parent, element.name)
7                       document_analysis(values.value, values.name)
8                   elseif element.istypeArray
9                       add_entity(values.name)
10                      add_relationship(parent, values.name)
11                      array_analysis(values.value, values.name)
12                  else
13                      add_entity_attribute(parent, values.name)
14                  end if
15          loop
16  end function
```

Code segment 7.5: Method: array_analysis

Table 7.2: Acceptability evaluation questions and code segments tested

QUESTION NUMBER	QUESTION	CODE TESTED
1	Does the artefact identify entities and attributes based on simulated data?	Code segment 7.1 Code segment 7.2 Code segment 7.3 Code segment 7.4 Code segment 7.5
2	Does the artefact identify possible primary keys based on simulated data?	Code segment 7.2
3	Does the artefact identify possible relationships based on simulated data?	Code segment 7.3 Code segment 7.4 Code segment 7.5
4	Does the artefact identify possible entities, attributes, keys and relationships based on simulated data, incorporating all four relational data model constructs?	All code segments are tested.

This study makes use of multiple samples of test data, which verifies that the artefact does what it is supposed to do. Each question is tested in all instances of the tests conducted using the various samples. The simulated data are loaded into the database; the artefact then analyses the data and presents the output; and the output is interpreted by the researcher. If all questions in Table 7.2 can be answered affirmatively then it will be concurred that the artefact is acceptable.

The following process outlines how each test is demonstrated:

1. Present data sample.
2. Demonstrate expected result from:
    a. the database designer's viewpoint, represented by an entity relationship diagram (ERD); and
    b. the programmer's viewpoint, represented by the expected output of the program.
3. List actual generated output of the program.

4. Analyse the results of steps 1 to 3 in order to answer the questions presented in Table 7.2.
5. Report results.

In each test, the four questions presented in Table 7.2 are addressed, but it should be noted that each test is motivated for specific purposes.

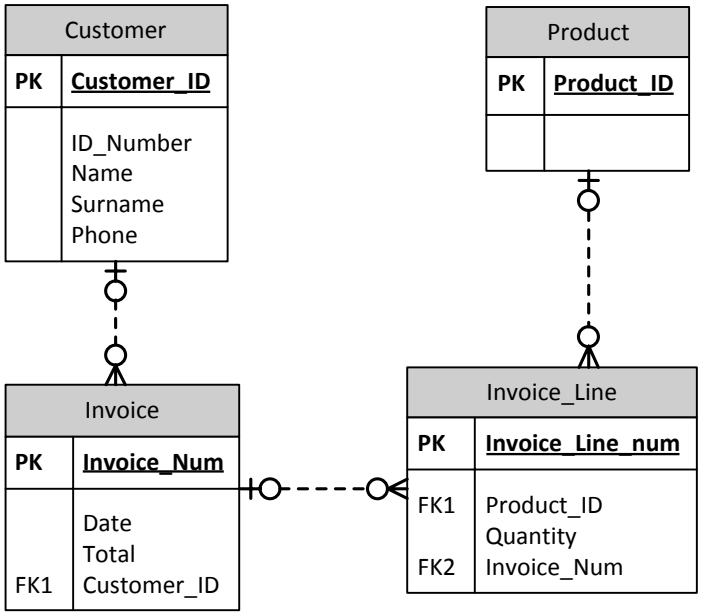
7.2.1 Test 1: Sample data from this study's running examples

Test 1 was conducted to demonstrate the artefact's analysis of the data. The data used for test 1 is the sample data used throughout this study. The sample data are related to the example data used in Chapter 3, Chapter 4 and Chapter 6. Figure 7.1 presents the sample data used for this test simulation. Table 7.3 shows the expected results of the analysis from the viewpoint of a database designer. This is followed by the expected output of the algorithm from the programmer's perspective. The actual output generated by the program is presented in Figure 7.2. Table 7.4 presents the analysis of the artefact's generated output using the questions from Table 7.2.


```
{
  "INVOICES" : [
    {
      "CUSTOMER" : {
        "CUSTOMER_ID" : "CUS101101",
        "ID_NUMBER" : "8802021243056",
        "NAME" : "Kelly",
        "PHONE" : "0169521122",
        "SURNAME" : "Mac Pearson"
      },
      "DATE" : "2012-01-02",
      "INVOICE_NUM" : "INV0012",
      "ITEMS" : [
        {
          "INVOICE_LINE_NUM" : "345",
          "PRODUCT_ID" : "POD1011",
          "QUANTITY" : "1"
        },
        {
          "INVOICE_LINE_NUM" : "346",
          "PRODUCT_ID" : "POD1012",
          "QUANTITY" : "100"
        },
        {
          "INVOICE_LINE_NUM" : "347",
          "PRODUCT_ID" : "POD2032",
          "QUANTITY" : "1"
        }
      ],
      "TOTAL" : "417-00"
    }
  ]
}
```


Figure 7.1: Sample data 1

Table 7.3: Expected viewpoints: Sample 1

VIEWPOINT	EXPECTED OUTCOME
DB DESIGNER	
PROGRAMMER	*Entities and Attributes:* customer (customer_id (pk) , id_number (pk) , name, phone, surname) invoices (date, invoice_num (pk), total) items (invoice_line_num (pk), product_id , quantity) *Relationships:* invoices-------<customer invoices-------<items

Entities and Attributes:

(_id (pk))

customer (customer_id (pk), id_number (pk), name (pk), phone (pk), surname (pk))

invoices (date (pk), invoice_num (pk), total (pk))

items (invoice_line_num (pk), product_id (pk), quantity)

Relationships:

-------<invoices

invoices-------<customer

invoices-------<items

Figure 7.2: Artefact-generated output: Sample 1

Table 7.4: Evaluation analysis: Sample 1

QUESTION	RESULT
1	Based on the output generated, all expected entities were identified, and no others were identified. The entities identified are “customer”, “invoices” and “items”. All expected attributes of the entities were identified, and no others were identified.
2	Based on the output generated, more primary keys were identified than expected. This is due to the fact that most fields only occur once and the limited number of records may be the reason for this. Primary keys were identified by the artefact, including the expected primary keys: “id_number”, “customer_id”, “name”, “phone” and “product_id”.
3	Based on the output generated, two expected relationships were identified between the entities “invoice”, “customer” and “items”.
4	Entities, attributes, primary keys and relationships were identified in the sample data.

The results in Table 7.4 were expected as the data sample is relatively simple to view and interpret. Various entities, attributes and relationships that were expected were identified. More than the expected primary keys were identified. From the researcher’s perspective, this output is realistic and the algorithm achieved the objective for the data of test 1.

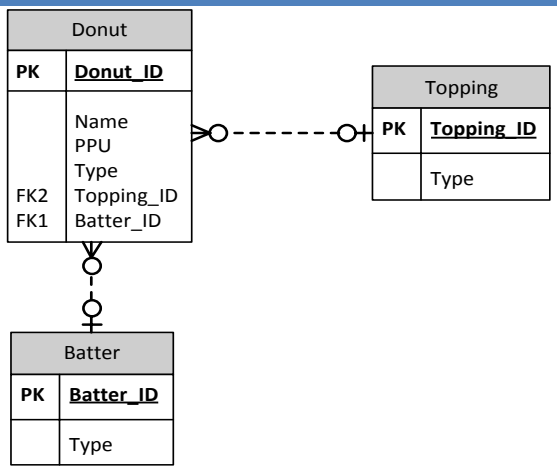
7.2.2 Test 2: Sample data from “JSON Data Set Sample” (Anon, 2013)

Test 2 was conducted to verify the test 1 results by using a different data set. This sample data set is relatively simple and was extracted from a web page named “JSON Data Set Sample” (Anon, 2013). This sample is based on recipes for donuts which include batters and toppings. Figure 7.3 presents the sample data used for this test simulation. Table 7.5 shows the expected results from the viewpoints of the database designer and programmer and are followed by the generated output of the program in Figure 7.4. Table 7.6 presents the analysis of the artefact’s generated output using the questions in Table 7.2.

```
{ "donuts": [
        {
                "id": "0001",
                "type": "donut",
                "name": "Cake",
                "ppu": 0.55,
                "batters":
                        {
                                "batter":
                                        [
                                                { "id": "1001", "type": "Regular" },
                                                { "id": "1002", "type": "Chocolate" },
                                                { "id": "1003", "type": "Blueberry" },
                                                { "id": "1004", "type": "Devil's Food" }
                                        ]
                        },
                "topping":
                        [
                                { "id": "5001", "type": "None" },
                                { "id": "5002", "type": "Glazed" },
                                { "id": "5005", "type": "Sugar" },
                                { "id": "5007", "type": "Powdered Sugar" },
                                { "id": "5006", "type": "Chocolate with Sprinkles" },
                                { "id": "5003", "type": "Chocolate" },
                                { "id": "5004", "type": "Maple" }
                        ]
        },
        {
                "id": "0002",
                "type": "donut",
                "name": "Raised",
                "ppu": 0.55,
                "batters":
                        {
                                "batter":
                                        [
                                                { "id": "1001", "type": "Regular" }
                                        ]
                        },
                "topping":
                        [
                                { "id": "5001", "type": "None" },
                                { "id": "5002", "type": "Glazed" },
                                { "id": "5005", "type": "Sugar" },
                                { "id": "5003", "type": "Chocolate" },
                                { "id": "5004", "type": "Maple" }
                        ]
        }
]
}
```


Figure 7.3: Sample data 2

Table 7.5: Expected viewpoints: Sample 2

VIEWPOINT	EXPECTED OUTCOME
DB DESIGNER	
PROGRAMMER	*Entities and Attributes:* (_id (PK)) batter id (PK), type) donuts (id (PK), name, ppu, type) topping (id (PK) , type) *Relationships:* batters-------<batter donuts-------<batters donuts-------<topping

Entities and Attributes:

(_id (PK))

batter (id, type)

batters ( )

donuts (id (PK), name (PK), ppu, type)

topping (id, type)

Relationships:

-------<donuts

batters-------<batter

donuts-------<batters

donuts-------<topping

Figure 7.4: Artefact-generated output: Sample 2

Table 7.6: Evaluation analysis: Sample 2

QUESTION	RESULT
1	Based on the output generated, most expected entities were identified, and one other was identified. The entities identified were “batter”, “donuts” and “toppings”. The other entity identified was “batters”, but whether this identified entity is an entity is debatable. Most expected attributes for the entities were identified, and no others were identified. However, the “batters” entity did not contain any attributes, which makes it debatable yet again whether it is an entity.
2	Based on the output generated, more primary keys were identified than expected. This is due to the fact that most fields only occur once and the limited number of records may be the reason for this. In some instances, primary keys that were expected to be identified were not identified, such as “id” in the “batter” entity and “id” in the “topping” entity. According to the data this is correct, since the “id = 1001” appears more than once in the data (refer to Figure 7.3). Primary keys were identified by the artefact, including the expected primary keys “id” and “name” of the “donuts” entity.
3	Based on the output generated, two expected relationships were identified between the entities “batter”, “donuts” and “toppings”. However, a third unexpected relationship was identified between “batters” and “batter”.
4	Entities, attributes, primary keys and relationships were identified in the sample data.

The results presented in Table 7.6 were expected as the data sample was relatively simple to view and interpret. Similar to test 1, various entities, attributes and relationships that were expected were identified. More than the expected primary keys were identified. From the researcher’s perspective, this output is realistic and the algorithm achieved the objective for the data of test 2.

7.2.3 Test 3: Modified sample data of test 2 from “JSON Data Set Sample” (Anon, 2013)

Test 3 was conducted to provide an explicit verification of primary key identification by modifying the data set used in test 2. The purpose of this test was to verify that the algorithm did what it is supposed to do, that is, execute the modified code segment 7.2. This sample was modified in order to incorporate the case where a primary key cannot be identified due to a value occurring more than once in the attribute. This was done by modifying the second record “id” attribute to the same value as the first record. Figure 7.5 presents the sample data used for this test simulation; the “id” is underlined to clearly illustrate the modification. Table 7.7 shows the expected results of the analysis from the viewpoint of a database designer. This is followed by the expected output of the algorithm from the programmer’s perspective. The actual output of the program generated is presented in Figure 7.6. Table 7.8 shows the analysis of the artefact’s generated output using the questions from Table 7.2.

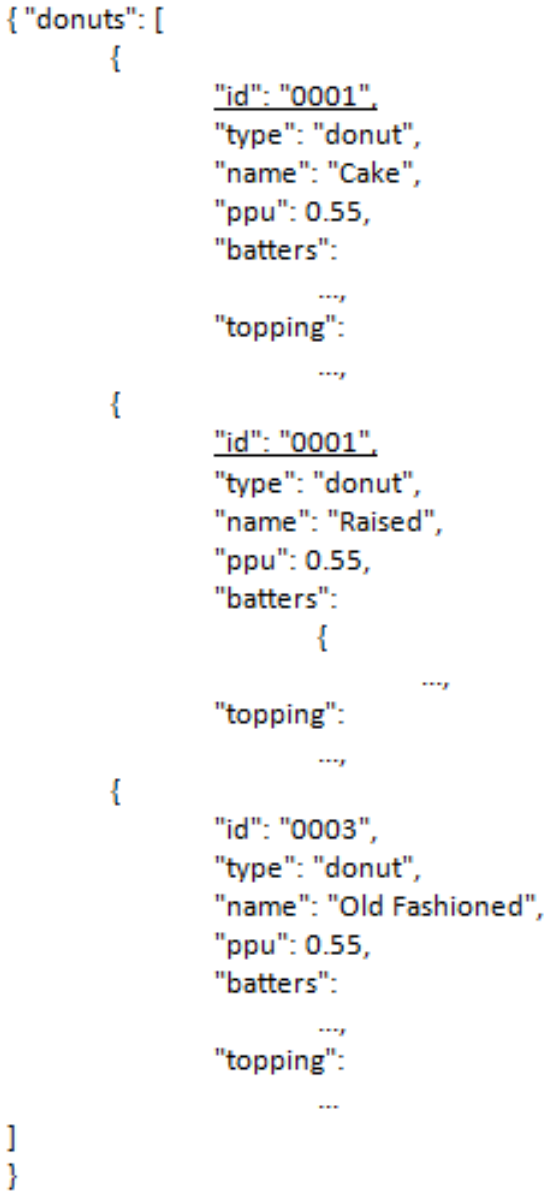


Figure 7.5: Sample data 3, modified version of sample data 2

Table 7.7: Expected viewpoints: Sample 3

VIEWPOINT	EXPECTED OUTCOME
DB DESIGNER	Refer to Table 7.5.
PROGRAMMER	*Entities and Attributes:* (_id (PK)) batter (id, type) donuts (id, name, ppu, type) topping (id, type) *Relationships:* batters-------<batter donuts-------<batters donuts-------<topping

Entities and Attributes:
(_id (PK))
batter (id, type)
batters ( )
donuts (id, name (PK), ppu, type)
topping (id, type)
Relationships:
-------<donuts
batters-------<batter
donuts-------<batters
donuts-------<topping

Figure 7.6: Artefact-generated output: Sample 3

Table 7.8: Evaluation analysis: Sample 3

QUESTION	RESULT
1	Same as in test 2.
2	Based on the output generated, it is clear that the (PK) indication for the "id" attribute in the "donuts" entity is no longer visible. This is due to the fact that the "id" attribute within this entity contains several of the same values. When comparing this result to the results generated in test 2, it is an indication that it is possible to identify possible primary keys within the data to a certain extent.
3	Same as in test 2.
4	Entities, attributes, primary keys and relationships were identified in the sample data.

The results in Table 7.8 were expected as the data sample was relatively simple to view and interpret. Similar to test 1 and test 2, various entities, attributes and relationships that were expected were identified. However, the previously expected and identified primary key from test 2 was no longer identifiable due the modified data. From the perspective of the researcher this output is realistic and the algorithm achieved the objective for the data of test 3.

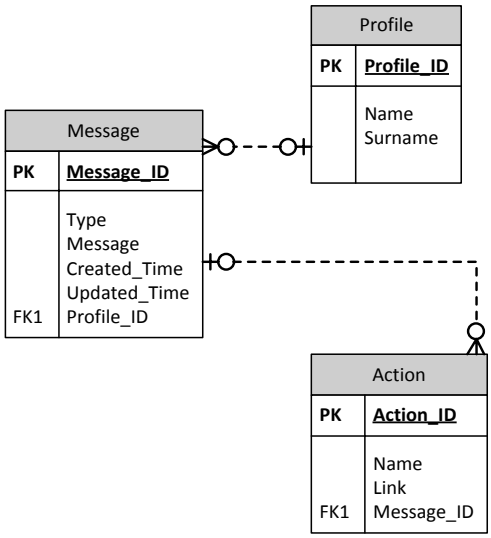
7.2.4 Test 4: Sample data from jQuery4u (Deering, 2011)

Test 4 was the last test conducted to provide final verification of the results of tests 1, 2 and 3, by using a third set of data totally different from the other data sets. This sample data set is relatively simple and was extracted from jQuery4u (Deering, 2011). This sample is based on details available about messages; it includes available comments made on messages and "likes" (similar to "likes" found on Facebook) for these messages. Figure 7.7 presents the sample data used for this test simulation. Table 7.9 shows the expected results of the analysis from the viewpoint of a database designer. This is followed by the expected output of the algorithm from the programmer's perspective. The actual output of the program generated is presented in Figure 7.8. Table 7.10 shows the analysis of the artefact's generated output using the questions from Table 7.8.

```
{
  "messages": [
    {
      "id": "X999_Y999",
      "from": {
        "name": "Tom Brady", "id": "X12"
      },
      "message": "Looking forward to 2010!",
      "actions": [
        {
          "name": "Comment",
          "link": "http://www.facebook.com/X999/posts/Y999"
        },
        {
          "name": "Like",
          "link": "http://www.facebook.com/X999/posts/Y999"
        }
      ],
      "type": "status",
      "created_time": "2010-08-02T21:27:44+0000",
      "updated_time": "2010-08-02T21:27:44+0000"
    },
    {
      "id": "X998_Y998",
      "from": {
        "name": "Peyton Manning", "id": "X18"
      },
      "message": "Where's my contract?",
      "actions": [
        {
          "name": "Comment",
          "link": "http://www.facebook.com/X998/posts/Y998"
        },
        {
          "name": "Like",
          "link": "http://www.facebook.com/X998/posts/Y998"
        }
      ],
      "type": "status",
      "created_time": "2010-08-02T21:27:44+0000",
      "updated_time": "2010-08-02T21:27:44+0000"
    }
  ]
}
```


Figure 7.7: Sample data 4

Table 7.9: Expected viewpoints: Sample 4

VIEWPOINT	EXPECTED OUTCOME
DB DESIGNER	
PROGRAMMER	*Entities and Attributes:* (_id (PK)) actions (link, name) messages (created_time, id (PK), message, type, updated_time) from (id, name) *Relationships:* messages -------<actions messages -------<from

Entities and Attributes:

(_id (PK))

actions (link, name)

messages (created_time, id (PK), message (PK), type, updated_time)

from (id (PK), name (PK))

Relationships:

-------<messages

messages -------<actions

messages -------<from

Figure 7.8: Artefact's generated output: Sample 4

Table 7.10: Evaluation analysis: Sample 4

QUESTION	RESULT
1	Based on the output generated, all expected entities were identified, and no others were identified. The entities identified are “messages”, “actions” and “from”. The “from” entity is unexplained but it could relate to the profile from where the message was sent. All expected attributes for the entities were identified, and no others were identified.
2	Based on the output generated, more primary keys were identified than expected. This is due to the fact that most fields only occur once and the limited number of records may be the reason for this. Primary keys were identified by the artefact, including the expected primary keys: “id”, “name” and “message”. “Message” was unexpected; however, when reflecting on it most messages are never the same.
3	Based on the output generated, two expected relationships were identified between the entities “messages”, “actions” and “from”.
4	Entities, attributes, primary keys and relationships were identified in the sample data.

The results in Table 7.10 were expected as the data sample was relatively simple to view and interpret. As in all the tests conducted, once again various entities, attributes and relationships that were expected were identified. More than the expected primary keys were identified. From the researcher’s perspective, this output is realistic and the algorithm achieved the objective for the data of test 4.

7.2.5 Conclusion of acceptability

Based on the interpretations of all the questions throughout Section 7.2.1 to Section 7.2.4, this study found that the artefact successfully addressed all the questions asked in Table 7.2. The researcher found that all the questions may be answered “Yes”. The researcher maintains that the artefact is acceptable and generated acceptable output.

Acceptability is the first evaluation criterion for the artefact. The researcher successfully addressed the first question: “Does the artefact present acceptable output based on the simulated data?” Table 7.11 is a summary of the questions and results based on the tests.

Table 7.11: Acceptability evaluation question results

QUESTION NUMBER	QUESTION	ADDRESSED	TESTED WITH
1	Does the artefact identify entities and attributes based on simulated data?	Yes	All tests.
2	Does the artefact identify possible primary keys based on simulated data?	Yes	Explicitly tested in test 3 with modified sample 3.
3	Does the artefact identify possible relationships based on simulated data?	Yes	All tests.
4	Does the artefact identify possible entities, attributes, keys and relationships based on simulated data, incorporating all four relational data model constructs?	Yes	All tests.

All the tests conducted yielded the expected results and the algorithm achieved the objective for the data. According to the researcher, the results are realistic and the artefact produces acceptable output. With the demonstration of acceptability of the artefact complete, this study continues with the evaluation of utility. The reader is reminded here of the constraints of the artefact in terms of many-to-many (M:N) relationships and faulty data discussed in Sections 6.4.4.2 and 6.5 respectively.

7.3 EVALUATION OF UTILITY

This section evaluates the artefact’s utility in the RAE. The RAE presented the researcher with data. These data were then processed using the artefact and the output was presented for interpretation. The generated output of the RAE was also evaluated by the researcher using the questions presented in Table 7.2. A representative from the RAE evaluated the generated output based on three main

questions. Table 7.12 lists the questions used to test the artefact. This table is used as a rubric of measurement for the artefact in the RAE.

Table 7.12: Utility in RAE evaluation questions

QUESTION NUMBER	THEME QUESTIONS	MOTIVATION
1	With the data presented, is the generated output acceptable?	Provide means of evaluation and validity.
2	With the data presented, does the generated output supply information that was not previously known?	Was the artefact successful in identifying all entities and attributes in the data?
3	With the data presented, is the generated output usable?	The newly discovered data are usable for the business.

Before the data could be generated for this study by the RAE, the RAE needed to adapt their data interception process; this process and its adaptation are discussed next.

7.3.1 Real application environment process adaptation

Newcom Fluid Management (NFM), the selected RAE for this study, uses one specific custom-built hardware-operated process. The data flow process starts with the fuel monitoring devices which send the data to an application server. The application server runs a protocol listener, enabling communication with field devices and the capturing of data generated by these devices. The listener decodes and formats the raw data to accommodate the current relational data structure.

For this study, NFM modified the listener to make a raw dump of the data to a NoSQL technology, namely MongoDB. The data were decoded and formatted for the relational data structure as usual, in order not to interrupt operations. As discussed in Section 5.2.2.3 it should be noted again that not all available data are utilised by the relational system. The data were exported by NFM and sent to be analysed. The procedure of how the data were imported for analysis by the artefact is available in

Appendix E. Figure 7.9 illustrates this process adaptation to accommodate this study.

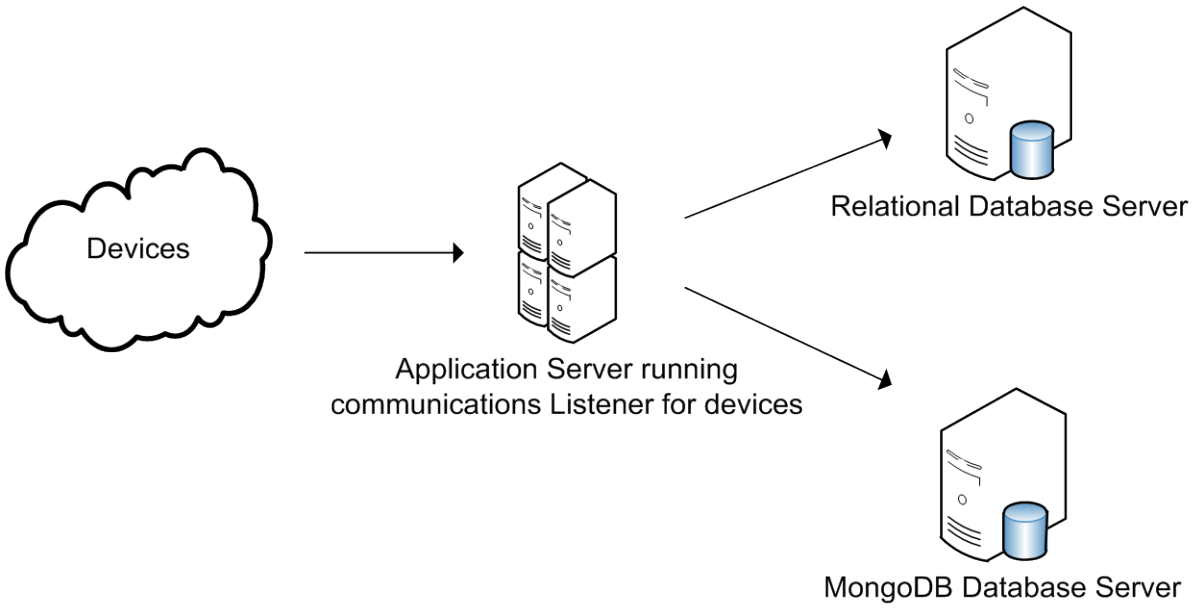


Figure 7.9: NFM process adaptation to accommodate the study

7.3.2 Real application environment test

The data presented by NFM consisted of an unknown number of records. Due to the magnitude of the data, it is not possible to show the actual data, and therefore a subset of these records was extracted and is presented in Appendix A. A second subset was extracted and is presented in Figure 7.10. The artefact, however, analysed all the records presented by NFM. The analysis of NFM's data using the artefact is subsequently presented similarly to the presentation of the data samples in Section 7.2.

Figure 7.10: Subset of NFM's data

Table 7.13 shows the viewpoints of a database designer and a programmer. The actual output generated by the program is presented in Figure 7.11. Table 7.14 gives the analysis of the artefact's generated output using the questions in Table 7.2.

Table 7.13: Viewpoints: sample NFM

VIEWPOINT	EXPECTED OUTCOME
DB DESIGNER	Impossible to determine due to the vast amount of data and the raw structure.
PROGRAMMER	Impossible to determine due to the vast amount of data and the raw structure.

Figure 7.11 presents the generated output of NFM's data. In the output generated, the attributes highlighted in bold illustrate newly discovered attributes not known by NFM. The NFM representative interpreted the output in the interview to determine usability as discussed in the next section.

Entities and Attributes:

(**(PK) , _id , 0x885054905220072C (PK) , 0x88753585B8837030 (PK) , AA87867272915956866 (PK) , AA885054905220072C (PK) , CA889976795D666735 (PK)** , capacity , **capture_type** , consumer_group_uid , consumer_type_uid , consumer_uid (PK) , current_hours , current_limit , **current_limit_date** , current_odometer , customer_uid , **DA877314887F946922 (PK) , DA87754675719B7047 (PK)** , daily_limit , description , km_limit , lk_cpk , max_kpl , min_kpl , name , prevoius_hours , prevoius_odometer , record (PK) , secondary_name , setup , tag)

deviceinfo (analog1 , analog2 , com1 , com2 , country , date (PK) , imei , ISO , manufacturer , patent , type , version)

events (consumer_uid , container_uid , customer_uid , data_event_uid , date (PK) , depot_uid , description , event_name , event_type , message , processed , record (PK) , station_uid , type , udatetime , worker_uid)

groups (consumer_group_uid, customer_uid, description, max_kpl, min_kpl, name, other, record)

packet ((PK))

readings (date, level)

status (analog1, analog2, com1, com2, date (PK) , imei , status , type)

transaction (auth_uid , **BHEX (PK) , BIN** , bypassed , consumer_transaction_uid (PK) , consumer_uid , cpk , date (PK) , **end_time** , hours , id , **imei** , kpl , location , odomenter , odometer , price , record (PK) , remarks , restype , **source** , **start_time** , trip_distance , udatetime , uid (PK) , **version** , volume , worker_uid)

transactions (date , id , odo , uid , volume)

types (consumer_type_uid , customer_uid , description , max_kpl , min_kpl , name , other , record)

Relationships:

-------<deviceinfo

-------<events

-------<groups

-------<packet

-------<status

-------<transaction

-------<types

transaction-------<events

transaction-------<readings

transaction-------<transactions

Figure 7.11: Artefact-generated output: sample NFM

Table 7.14: Evaluation analysis: sample NFM

QUESTION	RESULT
1	Based on the output generated, a list of 9 possible entities was identified. These entities are “deviceinfo”, “events”, “groups”, “packet”, readings”, status”, “transaction”, “transactions” and “types”. Extensive attribute lists were identified for each of the entities.
2	Primary keys were identified by the artefact. These keys include: “consumer_uid”, “record” and “uid”.
3	Based on the output generated, 10 possible relationships were identified between the entities.
4	Entities, attributes, primary keys and relationships were identified in the data. From the perspective of the researcher, this output is realistic and the algorithm achieved the objective.

The next section presents a data evaluation interview with a representative of NFM on the generated output of the artefact to support the first question in Table 7.12.

7.3.3 Evaluation of utility using real application environment data

An interview was conducted with a representative of NFM. The purpose of the interview was to:

a) Evaluate the generated output of the artefact, given the data presented by NFM; and
b) Evaluate the artefact’s utility in NFM.

A semi-structured interview was used to discuss the artefact with the representative of NFM. The themes of the questions for the interview are the same as those listed

in Table 7.12. The interview was transcribed and then analysed with regard to the evaluation of the artefact. The transcription of this interview is given in Appendix F.

The generated data in Figure 7.11 were presented to the NFM representative for evaluation. Table 7.15 gives the data analyses from the evaluation interview, using the qualitative data analysis technique discussed in Chapter 2. The first column is a coding value used to refer to the analysis. The second column is the overall question presented in Table 7.12. The third column expresses the quoted question asked by the interviewer and the response from the NFM representative, Mr Oosthuizen (2013c), and finally the fourth column is the theme identified.

Table 7.15: Qualitative data analysis of the evaluation interview.

CODE	THEME QUESTION	QUOTATION	THEME
Data:Known	From the data presented, is the generated output acceptable?	Interviewer: "Could you please indicate what existing data have you observed in the generated output?" Interviewee: "…quite a couple of parameters here, and to name but a few there would be capacity of the consumer, consumer groups, consumer types, consumer IDs, current operating hours or current limits, the descriptions, kilometre limits." Interviewee: "IDs again, the container IDs, the message, event names, descriptions." Interviewee: "Group ID, a description, the max kilometre-metres, name." Interviewee: "Bypassed status, the consumer transaction, the consumer ID number, the CDK information, the dates, the kilometre per litre, locations, odometers, to name but a few"	Known data
Data:Unknown	From the data presented, does	Interviewer: "Could you please indicate or elaborate a little bit on what existing- or what	Unknown

	the generated output present more data that was not previously known?	new or unknown data, or discarded data have you now also encountered in the data presented, in the generated output presented?" Interviewee: "… a couple of parameters which I don't really understand, which seemed to be still encoded…" Interviewee: "…there's some parameters here which I've newly discovered, and to name but a few again it's capture type." Interviewee: "There's actually device status information which is also fairly new." Interviewer: "Would you be able to use this data that was generated?" Interviewee: "Not only is there a lot of data captured that we currently use, there's also a lot of new data that's indicated that has been captured, which I think at this point can prove to be very valuable." Interviewer: "Could you please give an example?" Interviewee: "A good example would be to look at the start and end time of a transaction and determine the time spent within a day in transactions to determine where production gets lost due to ineffective refuel operations, and this might be due to laziness or misuse, and that certainly can contribute greatly to the overall operations, and within my organisation." Interviewee: "device status information is extremely powerful because you can do a lot of proactive maintenance by tracking and measuring your physical field hardware, and there find the stages, and in fact your quality of signal coming through is still intact or up to specifications, so that's extremely valuable data."	data

Busi:Usability	From the data presented, is the generated output usable?	Interviewer: “What value does this provide the business?” Interviewee: “Value would be quantified in terms of financial gain, providing new and more value-adding services to clients effectively, and also sustainably.” Interviewer: “Would the business like to expand on this research incorporating the artefact into the business information structure?” Interviewee: “It's an obvious choice from a business point of view to optimise your current resources, and yes, just looking at the current output I think there's such a big scope, so definitely.” “There's a lot of information that's new, and it would take time to process the information and actually verify the validity, but at first glance there's a lot of value, and I guess that's probably the name of the game. You know, it's continuous improvement, it's continuous growth, it's continuous evaluation, and you're only in front as long as you can run the fastest, so I think at this point this is one of the paths that we would like to venture [into] in the future to try and optimise our business as well.”	Usability
Busi:Unex	Miscellaneous information?	Interviewee: “How long would that be in the future for something like that if you have multiple sets of hardware platforms which communicate to a centralised data platform, and you have an artefact in this case which is effective in decoding multiple sets of data?” Interviewer: “The timeframe, is the research in such a form that it can be transferred quickly into a physical product or system, or do you still think there's still some time left for research?” Interviewee: “I have multiple sets of hardware in the field and I have my main database where	Unexpected

	my reports are generated from, and I'm going to put in this there now, this layer – which is this artefact which is going to eliminate all my drivers in the future?”

7.3.4 Conclusion of utility

In this section, the information from Table 7.15 is used to address the initial questions presented in Table 7.12.

The representative highlighted the fact that the artefact identified known attributes [Data:Known]. This was expected and is a means of self-evaluation for the representative to view the data as acceptable. Therefore when asked the question: “With the data presented, is the generated output acceptable?” The researcher may reason the answer to be “Yes”.

The business representative highlighted the fact that, in the generated output, the artefact generated unknown attributes [Data:Unknown]. This was the objective of the artefact. Therefore when asked the question: “With the data presented, does the generated output supply data that was not previously known?” The researcher may reason that the answer is “Yes”.

NFM responded that this data may be usable [Busi:Usability]. They could add more services to their client base [Busi:Usability]. The business representative believed that by “just looking, at first glance there's a lot of value” [Busi:Usability]. Therefore the answer to the question: “With the data presented, is the generated output usable?” May be reasoned to be “Yes” by the researcher.

The answers to all the questions in Table 7.12 were reasoned to be “Yes” by the researcher when provided with the evidence from NFM.

Acceptability of the artefact in a RAE was the second evaluation criterion for the artefact; this study can therefore conclude that the second question regarding evaluation, “Is the artefact usable for the RAE?” has been addressed successfully. The evidence for these answers given in the entire interview transcript can be found in Appendix F. Table 7.16 is a summary of the questions and answers.

Table 7.16: Utility of RAE evaluation results

QUESTION NUMBER	QUESTION	ADDRESSED	CODING REFERENCE
1	From the data presented, is the generated output acceptable?	Yes	Data:Known
2	From the data presented, does the generated output supply data that was not previously known?	Yes	Data:Unknown
3	From the data presented, is the generated output usable?	Yes	Busi:Usability

7.4 DEMONSTRATION AND EVALUATION SUMMARY

The evaluations of acceptability (Section 7.2) and utility (Section 7.3) confirm that the artefact is acceptable and usable. Therefore the artefact can analyse NoSQL document data stores to present possible entities, attributes, keys and relationships. Table 7.17 shows the main issues with questions answered for the evaluation and the evidence thereof.

Table 7.17: Main evaluation issues and their evidence

ISSUE	DESCRIPTION	ADDRESSED	EVIDENCE
1	Does the artefact present sufficient and acceptable output based on the simulated data?	Yes	See Section 7.3.2 and Tests 1 to 4.
2	Is the artefact usable for the RAE?	Yes	See Section 7.3.4, Appendix C and the RAE test.

This study used the information in Table 7.17 to support the following findings for the problem domains:

1. The generalised problem statement that data exist from various sources and in vast quantities that are unstructured and too complex to be viewed or interpreted by a human, has been addressed to some extent. The generalised problem statement has been solved within the document data store NoSQL data model.

2. The specialised problem statement that NFM has vast amounts of data that are unstructured and contain information that is hidden and not fully utilised by NFM to provide better services to their client base, has been addressed successfully.

This study found that the problem, in both the conceptual problem domain and the RAE problem domain, can be solved. The third finding of this study was beyond the initial expectation of the results. This finding was:

3. The RAE, that is NFM, was so interested in the artefact and research that they want to implement it as soon as possible. They asked questions concerning timeframes of implementations and whether the artefact would eliminate some of their drivers [Busi:Unex].

The third finding created the opportunity for a fifth design and development cycle and a new objective for the artefact was formulated:

The artefact needs to transfer selected entities and attributes to a RDBMS.

The graphical representation of the objective of the artefact was changed as illustrated in Figure 7.12.

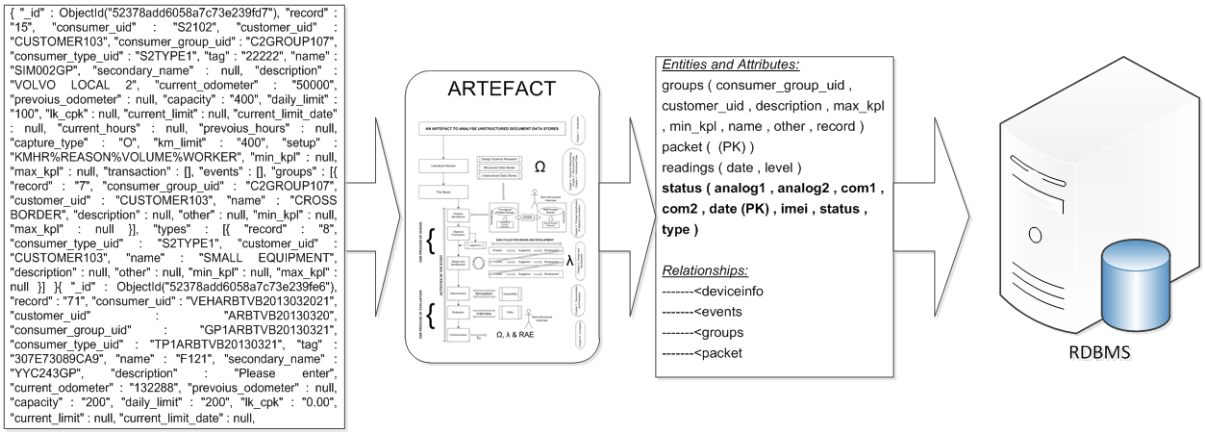


Figure 7.12: Processing objective of data by the artefact (modified)

The additional cycle proposed is briefly discussed next.

7.5 DESIGN AND DEVELOPMENT CYCLE BEYOND THE INITIAL DSR STUDY

In DSR, the analysis of performance assists to continuously improve the artefact. After the artefact had been evaluated by NFM, this DSR study found that NFM would benefit from incorporating the data identified by the artefact into their relational business systems. This need presented an unexpected fifth design and development cycle. This cycle is beyond the initial scope of this study and is briefly discussed here to demonstrate its feasibility. The fifth design and development cycle is used to demonstrate how it would be possible to transfer the data to a RDBMS which can then be utilised in a RAE, in conjunction with other structured data sources

7.5.1 Prescriptive knowledge specific to this design and development cycle

7.5.1.1 Globally Unique Identifier

A Globally Unique Identifier (GUID) is a 128-bit-long identifier that can guarantee a unique representation ID across space and time (Leach *et al.*, 2005). A GUID is also known as a Universally Unique Identifier (UUID) (Leach *et al.*, 2005).

In this study, GUID is used to uniquely group and identify records with their attributes and values.

7.5.2 Cycle 5: Transfer selected entities and attributes to a file or RDBMS

7.5.2.1 Problem and suggestions

It is pointless to transfer all the data that were identified by the artefact to a RDBMS, for it would create massive tables mostly filled with “null” values. This would defeat the objective of unstructured data stores. The researcher suggested that NFM should select entities and attributes to be transferred. Tables should be created for the selected entities and filled with the attributes and data. The creation of tables and the insertion of data into the RDBMS are done by generating SQL statements.

To enable this transfer of data, a few of the initial classes, methods and functions needed to be modified.

7.5.2.2 Development

Code segment 7.6 is an illustration of the modified *cls_Value* class that is used in the *cls_Attribute* class. This class is modified to store an extra list of record identifiers, *lst_record_uids*. The record identifiers are used to keep the record values grouped together. Figure 7.13 shows the UML class diagram for the *cls_Value* class.

```
1   class cls_Value
2           String var_content
3           Integer var_count
4           List lst_Record_uids
5   end class
```

Code segment 7.6: Class construct: cls_Value (modified)

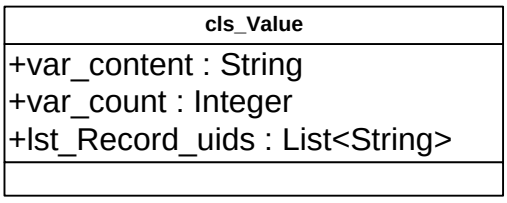


Figure 7.13: UML class diagram: cls_Value (modified)

Code segment 7.7 illustrates a modified *add_entity_attribute* function (CS7.7:1-27), which is the next step to exporting the data to a RDBMS.

```
1   function add_entity_attribute(name,  attribute, value, record_uid)
2       cls_Entity obj_currentEntity = lst_Entities(name)
3       if obj_currentEntity.lst_Attributes.contains(attribute)
4           cls_Attribute obj_currentAttribute = obj_currentEntity.lst_Attributes(attribute)
5           if currentAttribute.values.contains(value)
6               obj_currentAttribute.lst_Values(value).var_count =
7                   obj_currentAttribute.lst_Values(value).var_count + 1
8               obj_currentAttribute.lst_Values(value).lst_Record_uids.add(record_uid)
9                   //newly added line above
10              obj_currentEntity.lst_Attributes(attribute) = obj_currentAttribute
11          else
12              cls_Value obj_currentValue = new cls_Value()
13              obj_currentValue.value = value
14              obj_currentValue.lst_Record_uids.add(record_uid) //newly added line
15              obj_currentEntity.lst_Attributes(attribute).lst_Values.add(obj_currentValue)
16          end if
17      else
18          cls_Attribute obj_newAttribute = new cls_Attribute()
19          obj_newAttribute.var_name = attribute
20          cls_Value obj_newValues = new cls_Value()
21          obj_newValues.var_content = value
22          obj_newValues.var_count = 1
23          obj_newValues.lst_Record_uids.add(record_uid) //newly added line
24          obj_newAttribute.lst_Values = obj_newValues
25          obj_currentEntity.lst_Attributes.add(obj_newAttribute)
26      end if
27  end function
```

Code segment 7.7: Modified add_entity_attribute function for add record identifiers

In the function *add_entity_attribute* (CS7.7:1-27), a new parameter is added, namely "*record_uid*" (CS7.7:1). The variable "*record_uid*" is added to the *lst_Record_uids* list, in the *obj_newValues* (CS7.7:23) and *obj_currentValue* (CS7.7: and 14) of the *lst_Attributes* list, to group the values appropriately.

Code segment 7.8 introduces a new class *cls_Record* used in the artefact. The function of this class is to store the *var_record_uid, var_enity_name* and its attributes with their values. Figure 7.14 shows the UML class diagram for the *cls_Entity* class.

```
1   class cls_Record
2           String var_record_uid
3           String var_entity_name
4           List lst_Attributes_values
5   end class
```

Code segment 7.8: Class construct: cls_Record

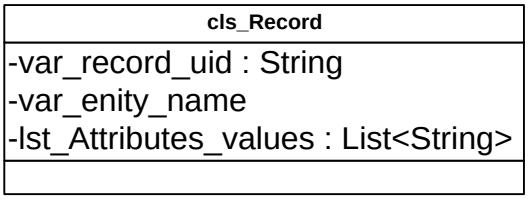


Figure 7.14: UML class diagram: cls_Record

Code segment 7.9 illustrates a modified *add_entity_attribute* function (CS7.9:1-21), which is the next step to exporting the data to a RDBMS. In the function *document_analysis* (CS7.9:2) and *array_analysis* (CS7.9:20), a new variable is added, namely *var_record_uid* (CS7.9:1). The variable *var_record_uid* is generated as a Globally Unique Identifier (GUID) to give it a unique value and the ability to group the attributes and values together. The variable *var_record_uid* is passed to the modified *add_entity_atribute* function.

```
1   function document_analysis(doc, parent)
2       String var_record_uid = newGUID() //New variable added
3       List lst_Elements = doc.elements
4       for each element in lst_Elements
5           if element.istypeDocument
6               add_entity(element.name)
7               add_relationship(parent, element.name)
8               document_analysis(element.value, element.name)
9           elseif element.istypeArray
10              add_entity(element.name)
11              add_relationship(parent, values.name)
12              array_analysis(element.value, element.name)
13          else
14              add_entity_attribute(parent, element.name, var_record_uid)
15          end if
16      loop
17  end function
18
19  function array_analysis(arr, parent)
20      String var_record_uid = newGUID() //New variable added
21      List lst_Values = arr.values
22      for each value in lst_Values
23          if element.istypeDocument
24              add_entity(values.name)
25              add_relationship(parent, element.name)
26              document_analysis(values.value, values.name)
27          elseif element.istypeArray
28              add_entity(values.name)
29              add_relationship(parent, values.name)
30              array_analysis(values.value, values.name)
31          else
32              add_entity_attribute(parent, values.name, var_record_uid)
33          end if
34      loop
35  end function
```

Code segment 7.9: Adding var_record_uid in the cls_Document_store_analyser class

Code segment 7.10 illustrates the *export* function (CS7.10:1-59), which is the final step to exporting the data to a text file or RDBMS. The function starts by declaring a two-string variable and a list variable (CS7.10:3-5), to which all other output is concatenated. A for-each statement is introduced, looping through all the entities in the *lst_Entities* list. If the entity is in the *lst_Selected_entities_attributes*, the algorithm continues. The *var_sql_entities_attributes* is concatenated with “CREATE TABLE”, the entity name and “RECORD_ID AUTO,” to create the SQL statement for the creation of the tables (CS7.10:9-10). The “AUTO” represents an auto number incremented by the RDBMS. Next, the algorithm loops through the attributes in the *lst_Attributes* of the entity. If the attribute is in the *lst_Selected_entities_attributes*, the algorithm continues. The *var_sql_entities_attributes* is concatenated with the attribute name and “TEXT” to create the SQL statements for the creation of the attributes in the tables (CS7.10:13-14). Next, the algorithm loops through the values in the *lst_Values* of the attribute and then through the “uid” in the *lst_Record_uids list*. The *lst_Records* list is checked to ascertain whether it already contains the “uid” (CS7.10:17). If the list contains the value, then the current *cls_Record* object in the list is updated with the new attribute and its value (CS7.10:17-22). If the list does not contain the value then a new object is created of the *cls_Record* class, the attribute and its values are added to this object and the object is added to the *lst_Records* list. (CS7.10:20-25). Finally, the *var_sql_entities_attributes* is concatenated again with “)”.

The algorithm continues to loop thought the previously generated *lst_Records* list. The variable *var_sql_data* is concatenated with “INSERT INTO” (CS7.10:42), the entity name and “(”. The algorithm now loops through each attribute/value pair and concatenates the variable *var_sql_data* with the attribute name and “,”. The variable *var_sql_data* is concatenated then with “) VALUES (`”. The algorithm loops through each attribute/value pair again and concatenates the variable *var_sql_data* with the attribute value and “`,`”. Finally the variable *var_sql_data* is concatenated with “`)”.

The “*var_sql_entities_attributes”* and the *“var_sql_data”* can now be written to a file or executed against a RDBMS (CS7.10:57-58).

```
1   function export()
2
3       List lst_Records
4       String var_sql_entities_attributes
5       String var_sql_data
6
7       for each entity in lst_Entities
8           if lst_Selected_entities_attributes.contails(entity.var_name)
9                   var_sql_entities_attributes  = var_sql_entities_attributes  +
10                      "CREATE  TABLE  " + entity.var_name  + "( RECORD_ID  AUTO ,"
11                  for each attribute in lst_Attributes
12                      if lst_Selected_entities_attributes.contails(attribute.var_name)
13                          var_sql_entities_attributes  = var_sql_entities_attributes  +
14                              attribute.var_name  + " TEXT,  "
15                          for each value in attribute.lst_Values
16                              for each uid in value.lst_Record_uids
17                                  if lst_records.contain(uid)
18                                      cls_Record  obj_currentRecord  = lst_Records(uid)
19                                      obj_currentRecord.lst_Attributes_values.add(
20                                          attribute.var_name,  value.var_content)
21                                      lst_Records(uid)  = obj_currentRecord
22                                  else
23                                      cls_Record  obj_newRecord  = new cls_Record()
24                                      obj_newRecord.var_record_uid  = uid
25                                      obj_newRecord.var_entity_name  =
26                                          obj_currentEntity.var_name
27                                      obj_newRecord.lst_Attributes_values  = New List()
28                                      obj_newRecord.lst_Attributes_values.Add(
29                                          attribute.var_name,  value.var_content)
30                                      lst_Records.Add(uid,  obj_newRecord)
31                                  end if
32                              loop
33                          loop
34                      end if
35                  loop
36                  //remove last 2 characters with subString from var_sql_entities_attributes
37                  var_sql_entities_attributes  = var_sql_entities_attributes  + ")" + NewLine
38          end if
39      loop
```

```
40
41          for each record in lst_Records
42              var_sql_data = var_sql_data + "INSERT INTO" + record.var_entity_name + "("
43              for each attribute_value in lst_Records
44                  var_sql_data = var_sql_data + record.lst_Attributes_values.GetKey(
45                      attribute_value) + " , "
46              loop
47              //remove last 2 characters with subString from var_sql_entities_attributes
48              var_sql_data = var_sql_data + ") VALUES (`"
49              for each attribute_value in lst_Records
50                  var_sql_data = var_sql_data + record.lst_Attributes_values.GetValue(
51                      attribute_value) + "` , `"
52              loop
53              //remove last 2 characters with subString from var_sql_entities_attributes
54              var_sql_data = var_sql_data + "`)"
55          loop
56
57          // write var_sql_entities_attributes and var_sql_data to a file OR
58          // execute var_sql_entities_attributes and var_sql_data on a RDBMS server
59      end function
```

Code segment 7.10: Export function to export data

Using the initial example of this study, Figure 7.1, a few entities and attributes were selected (listed in Table 7.18) and added to the *lst_Selected_entities_attributes list.*

Table 7.18: Selected entities and their attributes

ENTITY	ATTRIBUTES
CUSTOMERS	CUSTOMER_ID, ID_NUMBER, NAME, PHONE
INVOICES	DATE, INVOICE_NUM, TOTAL
ITEMS	INVOICE_LINE_NUM, PRODUCT_ID, QUANTITY

Figure 7.15 demonstrates the SQL statements generated by the algorithm for selected entities and attributes in Table 7.18. These statements may be executed in a RDBMS with other data.

CREATE TABLE CUSTOMER(RECORD_UID AUTO,CUSTOMER_ID TEXT, ID_NUMBER TEXT, NAME TEXT, PHONE TEXT)

CREATE TABLE INVOICES(RECORD_UID AUTO,DATE TEXT, INVOICE_NUM TEXT, TOTAL TEXT)

CREATE TABLE ITEMS(RECORD_UID AUTO,INVOICE_LINE_NUM TEXT, PRODUCT_ID TEXT, QUANTITY TEXT)

INSERT INTO ITEMS(INVOICE_LINE_NUM , PRODUCT_ID , QUANTITY) VALUES (`346` , `POD1012` , `100 `)

INSERT INTO ITEMS(INVOICE_LINE_NUM , PRODUCT_ID , QUANTITY) VALUES (`347 `, `POD20321 , `1`)

INSERT INTO ITEMS(INVOICE_LINE_NUM , PRODUCT_ID , QUANTITY) VALUES (`345` , `POD101`1 , `1`)

INSERT INTO CUSTOMER(CUSTOMER_ID , ID_NUMBER , NAME , PHONE) VALUES (‘CUS101101` , `8802021243056` , KELLY , `0169521122`)

INSERT INTO INVOICES(DATE , INVOICE_NUM , TOTAL) VALUES (`2012-01-02` , `INV0012` , `417-00`)

Figure 7.15: Generated SQL statements for selected entities and attributes

Evaluation of the generated SQL code from Figure 7.15 showed that the handling of relationships remains a major limitation to the artefact and may be the subject of future research.

Figure 7.16 shows the UML class diagram for the modified *cls_Document_store_analyser* class, the relationships with the original classes and the new *cls_Record* class.

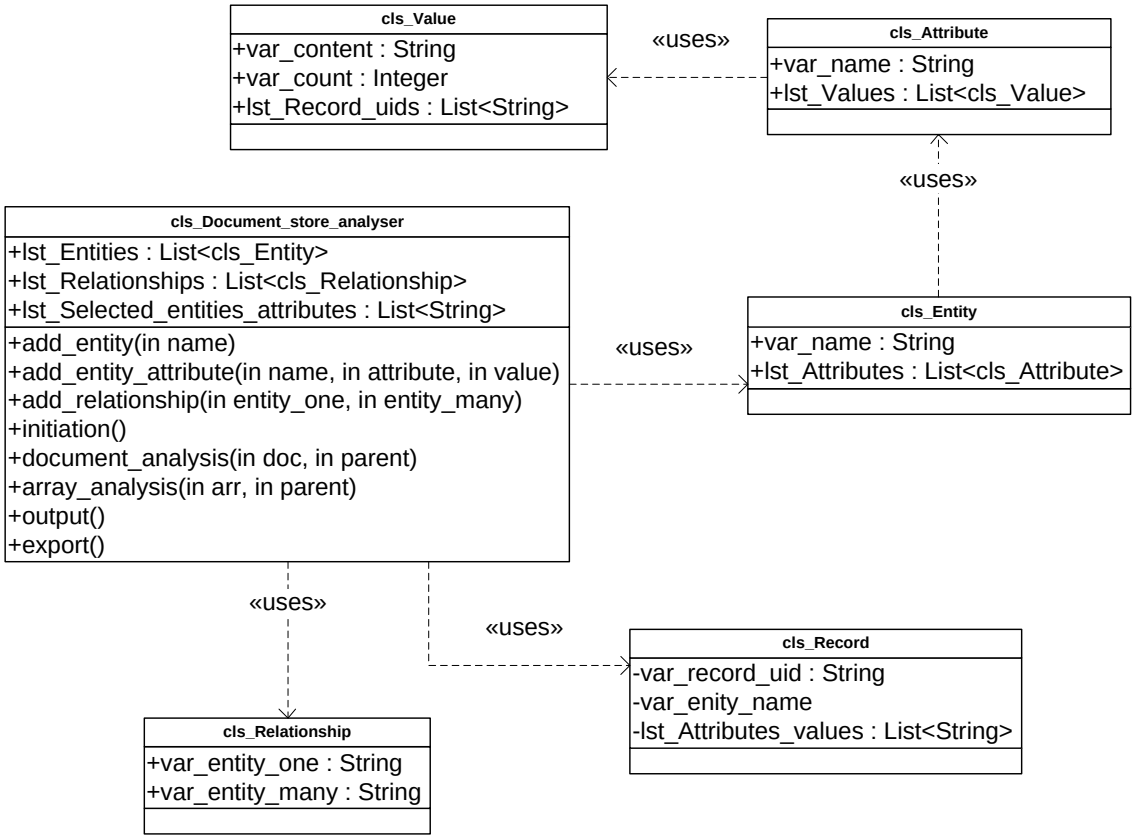


Figure 7.16: UML class diagram: cls_Document_store_analyser (modified iteration)

Further demonstration and evaluation of the modified algorithm remain beyond the scope of this study. Figure 2.7 in Chapter 2, which guided the researcher through the study, has been revisited and is illustrated in Figure 7.17, indicating the possibility of future research.

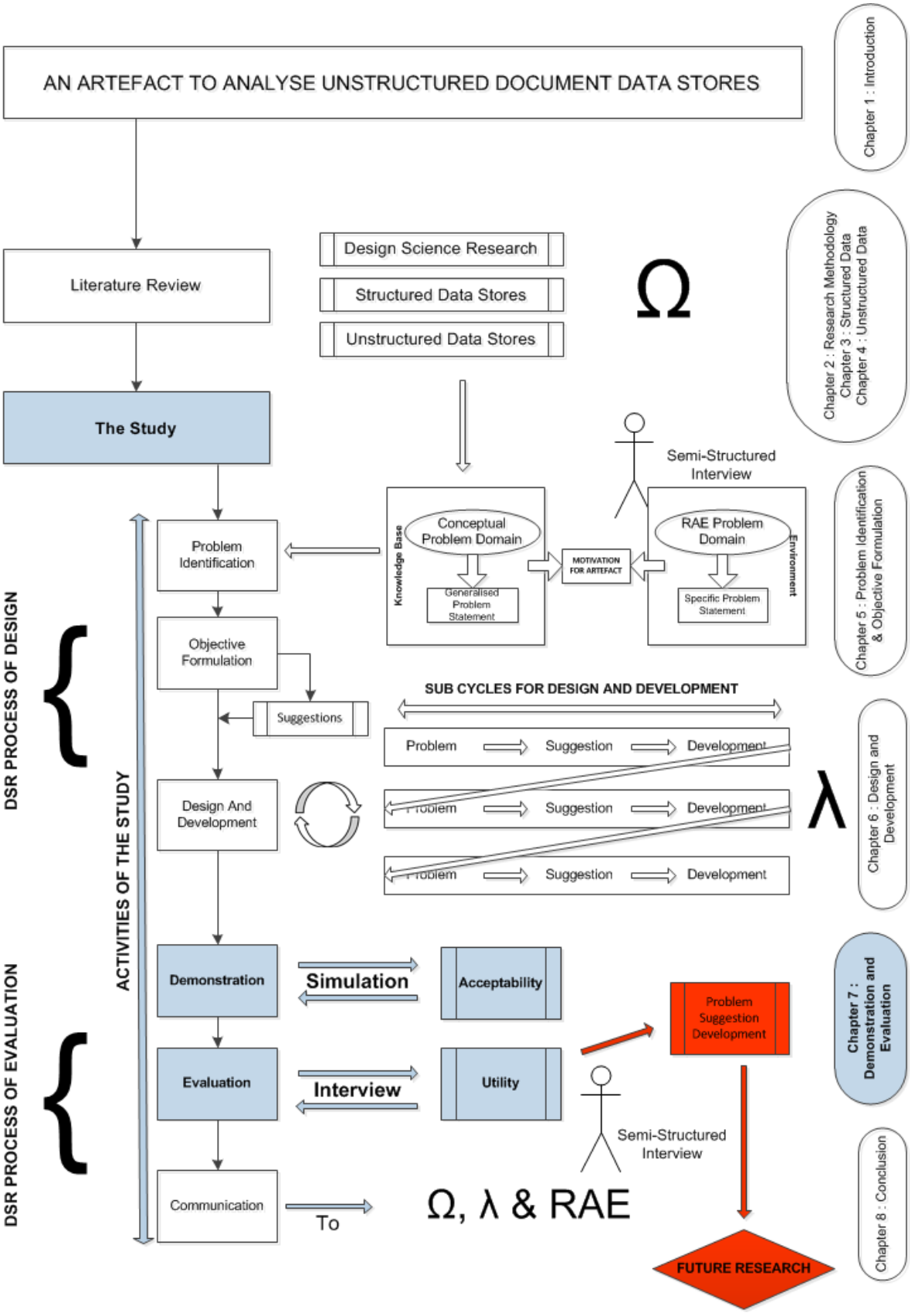


Figure 7.17: Illustration of the process followed in this study (revisited)

7.6 CHAPTER CONCLUSION

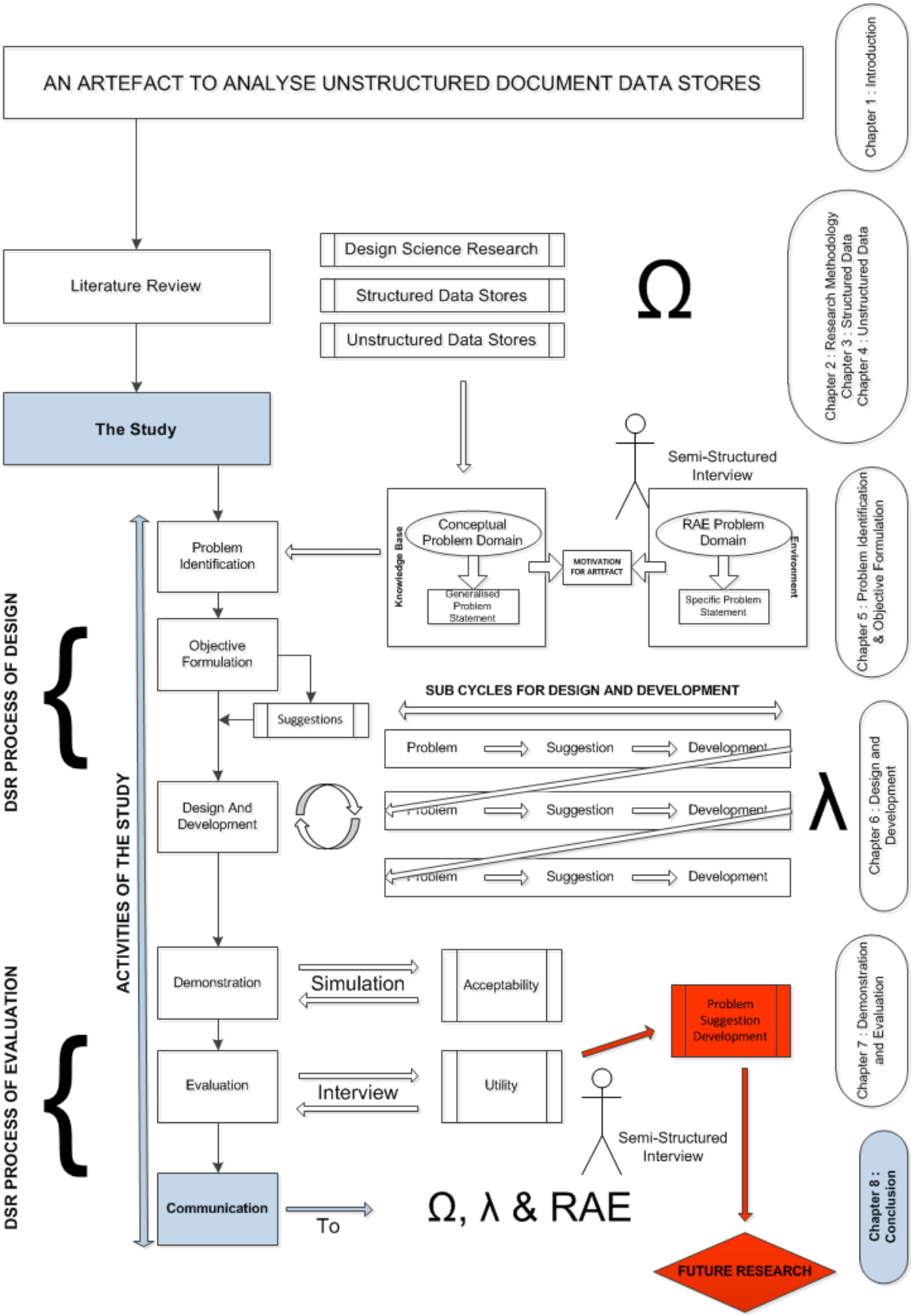
The objective of this chapter was to report on the evaluation of the artefact's acceptability to the researcher and its utility within the RAE. This objective was met by evaluating and demonstrating the acceptability of the artefact through the use of tests. The utility of the artefact was evaluated in the RAE and evidence is provided in Appendix F.

The evaluation of the artefact provided positive results and the researcher can conclude that it is possible to analyse document data stores using relational data model constructs.

The objective of the artefact presented in Chapter 5, that is: The artefact needs to analyse the document data store and identify entities, attributes and relationships, has been addressed successfully. The artefact can identify entities, attributes and relationships in document data store, presenting acceptable and interpretable output for the user.

It was also demonstrated how the artefact could be extended to allow for integration of the document data store data with structured data in an organisation.

The next chapter concludes and communicates this DSR study.


AN ARTEFACT TO ANALYSE UNSTRUCTURED DOCUMENT DATA STORES
Literature Review
The Study
Design Science Research
Structured Data Stores
Unstructured Data Stores
Ω
Chapter 1 : Introduction
Chapter 2 : Research Methodology
Chapter 3 : Structured Data
Chapter 4 : Unstructured Data
Semi-Structured Interview
Problem Identification
Objective Formulation
Suggestions
Design And Development
Demonstration
Evaluation
Communication
Knowledge Base
Conceptual Problem Domain
Generalised Problem Statement
MOTIVATION FOR ARTEFACT
RAE Problem Domain
Specific Problem Statement
Environment
Chapter 5 : Problem Identification & Objective Formulation
SUB CYCLES FOR DESIGN AND DEVELOPMENT
Problem
Suggestion
Development
λ
Chapter 6 : Design and Development
DSR PROCESS OF DESIGN
ACTIVITIES OF THE STUDY
DSR PROCESS OF EVALUATION
Simulation
Acceptability
Interview
Utility
Problem Suggestion Development
Semi-Structured Interview
Chapter 7 : Demonstration and Evaluation
To
Ω, λ & RAE
FUTURE RESEARCH
Chapter 8 : Conclusion

CHAPTER EIGHT: COMMUNICATION

8.1 INTRODUCTION

The primary objective of this study was to develop an artefact which analyses document data stores in terms of structured data model constructs. In order to achieve this, the study formulated a design science research (DSR) methodology, discussed in Chapter 2. This chapter presents the final activity of this study's DSR methodology namely, communication.

In the context of this DSR study, communication is the summary of what has been learned from this DSR study. It is also a means of self-reflection using the questions (refer to Table 2.6, duplicated in Table 8.1 in Section 8.2.2) provided by Hevner and Chatterjee (2010:20).

The objective of this chapter is to communicate and conclude this DSR study. The chapter summarises all the key concepts from the previous chapters, answering the research question and addressing the objectives of the study.

This chapter is divided into the following sections: summary of research findings of the study (Section 8.2); recommendations for future research (Section 8.3); and finally, the conclusion (Section 8.4).

8.2 RESEARCH FINDINGS OF THE STUDY

This section revisits the research question, the primary objective and the theoretical objectives by providing information on how these were addressed in this DSR study.

The research question for this study is:

> *Is it possible to design and develop an artefact that could analyse NoSQL document data stores in terms of relational data model constructs?*

This research question is supported by the primary objective:

> *The primary objective of this study is to develop an artefact which analyses document data stores in terms of structured data model constructs.*

The primary objective is supported by these theoretical objectives:

4. Gain an understanding of design science research.
5. Gain an understanding of structured data stores by focusing on the relational data model, its constructs and its characteristics.
6. Gain an understanding of unstructured data stores, such as NoSQL, by focusing on document data stores, their constructs and their characteristics.

The following sections reflect on the research question, the primary objective and theoretical objectives by highlighting key findings.

8.2.1 Theoretical objectives

The sections that follow represent key findings based on the literature review during this DSR study.

8.2.1.1 Design science research

The design and development of an artefact is an important requirement of the primary objective and theoretical objectives. DSR methodology was explained through a review of the existing literature in Chapter 2 – addressing the first theoretical objective.

This study found that DSR is a problem-solving research paradigm, where the researcher is interested in gaining knowledge through creating. The value of the artefact is important to the study; therefore, the philosophical position taken is that of DSR. The control, creation and understanding of the designed artefact are important to the researcher regarding this study.

DSR represents the understanding of a problem domain, where the development and evaluation of an artefact provide a solution. Various DSR methodologies are available for a DSR project, and sometimes these methodologies can be combined to present a better research methodology. The combined DSR methodologies of Peffers *et al.* (2008) and Vaishnavi and Kuechler (2004) provided a sound structure for this study. A diagram was developed to guide the research activity and was provided at the start of each chapter.

8.2.1.2 Structured data stores

An understanding was gained of the relational data model and RDBMSs through a review of the existing literature in Chapter 3 – addressing the second theoretical objective.

The relational data model, built on the mathematical concept known as a relation, has been the dominating data model for the past few decades. The relational data model is a set of tables, which are related to one another through a common set of attributes. The relational data model is a structured data store that may be graphically represented by a logical view. These graphical representations allow visual views of complex real-word database structures.

The relational data model is implemented by a relational database management system (RDBMS). A RDBMS is software designed to utilise large collections of data while hiding complexities from the user.

This study found that the relational data model is an environment that presents a simple logical view of data which is easy for a human to interpret.

8.2.1.3 Unstructured data stores

An understanding was gained on the document data store model and NoSQL DBMSs through a review of the existing literature in Chapter 3 – addressing the third theoretical objective.

NoSQL is a non-relational, distributed, open-source and horizontally scalable DBMS. It evolved from the need for better horizontal scalability regarding the distribution of data over multiple servers. It addresses the need to store unstructured data that is represented by four data models: key-value stores, document stores, column-oriented stores and graph databases.

The CAP theorem states that only two of the three properties (consistency, availability and partition tolerance) can be satisfied at any given time. This theorem is the basis for all NoSQL DBMSs. NoSQL DBMSs all incorporate two of these properties in some way or another.

Scalability (vertical, horizontal or sharding) is implemented to allow databases to be distributed over many computers.

MapReduce enables parallel processing of large data sets. It enables program execution to become distributed across several machines without the programmer having to worry about the implementations.

Multi-version concurrency control (MVCC), one of the concurrency control strategies, enables simultaneous access to a database. It prevents data corruption and deadlocks by locking the data and not allowing any other process to access the data.

Optimistic locking, another concurrency control strategy, checks for conflicts and only rolls back the data when conflicts are found.

This study found that the NoSQL data models mostly store unstructured and denormalised data needed for Big Data.

8.2.1.4 NewSQL

From the review of the existing literature, NewSQL was found to present itself as an alternative to NoSQL. NewSQL is important to take note of as it provides further motivation for the development of an artefact which enables the transformation of data from unstructured stores to these new structured data stores.

NewSQL refers to a new generation of RDBMS that could accommodate performance in scalability, similar to those found in NoSQL, while incorporating the classic ACID properties (Moniruzzaman & Hossain, 2013).

NewSQL supports the common relational data model and makes use of SQL as a query language, but their architectures may differ (Cattell, 2011:20).

This study found that unstructured data captured with NoSQL remains important, but eventually organisations may want to transform such data into a structured format and NewSQL could support this.

8.2.2 Primary objective: Development of the artefact

The sections that follow discuss key findings concerning the development of the artefact in this DSR study which addresses the primary objective.

The DSR approaches of Peffers *et al*. (2008) and Vaishnavi and Kuechler (2004) were integrated to devise a structure for this study. The Vaishnavi and Kuechler (2004) approach makes use of inner cycles for development and forms the structural support for the design, while the development activity is informed by the Peffers *et al*. (2008) approach.

The questions in Table 8.1 provided by Hevner and Chatterjee (2010:20) constitute a checklist for self-evaluation of this DSR study. In the text of some sections, Table 8.1 is referred to in order to address the specific question.

Table 8.1: This DSR study's self-reflection checklist

NUMBER	QUESTION	SECTION ADDRESSED
1	What is the research question (design requirements)?	8.2
2	What is the artefact? How is the artefact represented?	8.2.2.2
3	What design processes (search heuristics) will be used to build the artefact?	8.2.2.2
4	How are the artefact and the design processes grounded by the knowledge base? What, if any, theories support the artefact's design and the design process?	8.2.2.2 is supported by 8.2.1.2 and 8.2.1.3
5	What evaluations are performed during the internal design cycles? What design improvements are identified during each design cycle?	8.2.2.3
6	How is the artefact introduced into the application environment and how is it field tested?	8.2.2.3
7	What metrics are used to demonstrate artefact utility and improvement over previous artefacts?	8.2.2.3
8	What new knowledge is added to the knowledge base and in what form (e.g., peer-reviewed literature, meta-artefacts, new theory and new method)?	8.2.3 and 8.3
9	Has the research question been satisfactorily addressed?	8.2.2.2 and 8.4

8.2.2.1 Problem and objective formulation

In Chapter 5, the problem and objective formulation activities presented the first and second activities of this DSR study. Hevner and Chatterjee (2010) and Iivari (2007)

state that DSR begins in the application environment when an opportunity, challenging problem or insightful vision for something innovative is presented. However, this was not the case for this study. The main origin of the problem presented itself through the review of the literature. The RAE was only addressed at a later stage in order to support the artefact's importance. Therefore the problem domain was obtained from both a conceptual view and the RAE.

The conceptual problem domain provided the following problem statement through the review of the literature:

> *Data exist from various sources and in vast quantities that are unstructured and too complex to be viewed or interpreted by a human.*

The RAE problem domain provided the following problem statement through an interview:

> *NFM has vast amounts of data that are unstructured and contain information that is hidden and not fully utilised by NFM to provide better services to their client base.*

The conceptual problem domain statement provided the primary motivation for the development of the artefact, while the RAE problem domain statement demonstrated the value for the artefact.

This study found that a DSR project does not necessarily start within a RAE as stated by Hevner and Chatterjee (2010) and Iivari (2007). A problem domain may present itself to a researcher through a review of the existing literature.

Based on the problem statements, the objective for the artefact formulated is:

The artefact needs to analyse the document data store in order to identify entities, attributes and relationships.

Figure 8.1 illustrates the objective of the artefact where the artefact needs to collect data, process it and present output.

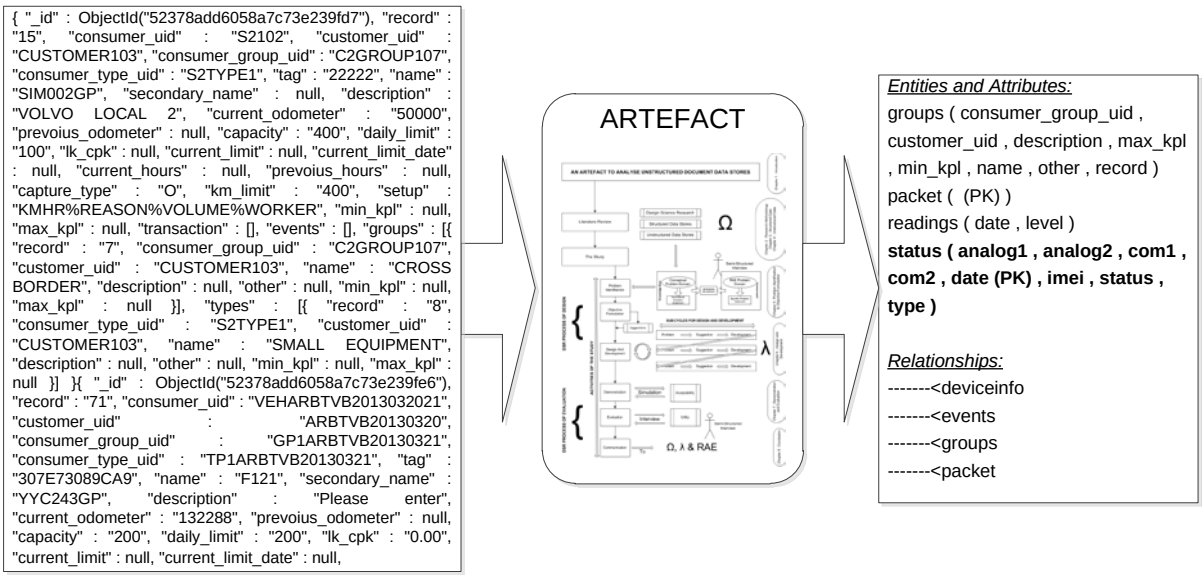


Figure 8.1: Processing objective of data by the artefact (initial)

The objective of the artefact was supported by suggestions made by the researcher. The suggestions were implemented in the design and development activity of the study and are discussed next.

8.2.2.2 Artefact design and development

The design and development activity presented the third activity of this DSR study. The design and development of the artefact was discussed in Chapter 5.

Regarding question 2 (refer to Table 8.1), it is evident that the artefact is an implementation of an algorithm that analyses document data stores to identify possible entities, attributes, primary keys and relationships. The identification takes place based on relational model constructs. It is presented in the form of pseudocode to describe the steps within the algorithm. The algorithm implemented object-oriented design concepts and continues design concepts to improve the artefact. The complete initial algorithm is available in Appendix C and the actual code implementation is available in Appendix D.

The knowledge used to develop the algorithm with regard to question 4 (refer to Table 8.1) draws from the literature, that is, descriptive knowledge explored in Chapters 2 and 3. The literature discusses the relational model constructs that are drawn from the relational model in order to identify possible entities, attributes, primary keys and relationships. The literature also covers NoSQL, specifically the document data model, and how it works.

Regarding question 3 (refer Table 8.1), it is stated that the artefact was designed by implementing the process of continuous design. The artefact was initially designed, developed and then improved with each cycle of the design. This led to the final design which incorporated all the suggestions made in this study as discussed in Chapter 5.

This study found that an artefact can assist with the analyses of unstructured data stores and presents a logical view of the data without having to browse millions of records. This addressed the main research question of this study referring to question 9 in Table 8.1.

8.2.2.3 Demonstration and evaluation

In Chapter 7, the demonstration and evaluation activities presented the fourth and fifth activities of this DSR study. The study formulated two criteria for evaluation. The first criterion is acceptability, and the second criterion is utility. The two criteria for

evaluation are the main issues to be addressed by this section and are listed in Table 8.2.

Table 8.2: Main evaluation issues

ISSUE	DESCRIPTION	VALUE	ADDRESSED	EVIDENCE
1	Does the artefact present sufficient and acceptable output based on the simulated data?	This demonstrates the artefact's value in terms of quality.	Yes	See Section 7.3.2
2	Is the artefact usable in the RAE?	This demonstrates the artefact's value in terms of utility.	Yes	See Section 7.3.4 and Appendix F.

The first issue, demonstrating acceptability, was addressed using tests and simulated data with reference to questions 5 and 7 in Table 8.1. The criteria used for the tests are given in Table 8.3.

With regard to questions 6 and 7 (refer Table 8.1), the second process that evaluates utility was carried out using an interview in the RAE. The RAE was presented with output generated by the artefact. The RAE was asked to evaluate this output and interpret it, thereby giving feedback about the artefact. The criteria used for the RAE are illustrated in Table 8.4 and Section 7.3.3, which provides substantial evidence thereof.

Table 8.3: Acceptability evaluation results

QUESTION NUMBER	QUESTION	ADDRESSED	TESTED WITH
1	Does the artefact identify entities and attributes based on simulated data?	Yes	All tests, Section 7.2.
2	Does the artefact identify possible primary keys based on simulated data?	Yes	Explicitly tested in test 3 with modified sample 3, Section 7.2.4.
3	Does the artefact identify possible relationships based on simulated data?	Yes	All tests, Section 7.2.
4	Does the artefact identify possible entities, attributes, keys and relationships based on simulated data, incorporating all four relational data model constructs?	Yes	All tests, Section 7.2.

Table 8.4: Utility of RAE evaluation results

QUESTION NUMBER	QUESTION	ADDRESSED	CODING REFERENCE
1	From the data presented, is the generated output acceptable?	Yes	Data:Known
2	From the data presented, does the generated output supply data that were not previously known?	Yes	Data:Unknown
3	From the data presented, is the generated output usable?	Yes	Busi:Usability

For the purposes of this study, proof of concept is sufficient, and therefore the artefact is not fully developed to be field tested.

This study found that the artefact solved a subset of the conceptual problem domain and solved the RAE problem domain. The demonstration and evaluation of the artefact, as discussed in Chapter 7, provided sufficient evidence in this regard.

An unexpected result obtained during the evaluation of the artefact in the RAE was that the RAE user was eager to implement the artefact in their structured business environment. This guided the study towards an unexpected fifth design and development cycle of the artefact, which was beyond the scope of the initial study. A new objective of the artefact was formulated:

The artefact needs to transfer selected entities and attributes to a RDBMS.

The picture of the objective of the artefact changed slightly due to the new objective formulated and is illustrated in Figure 8.2.

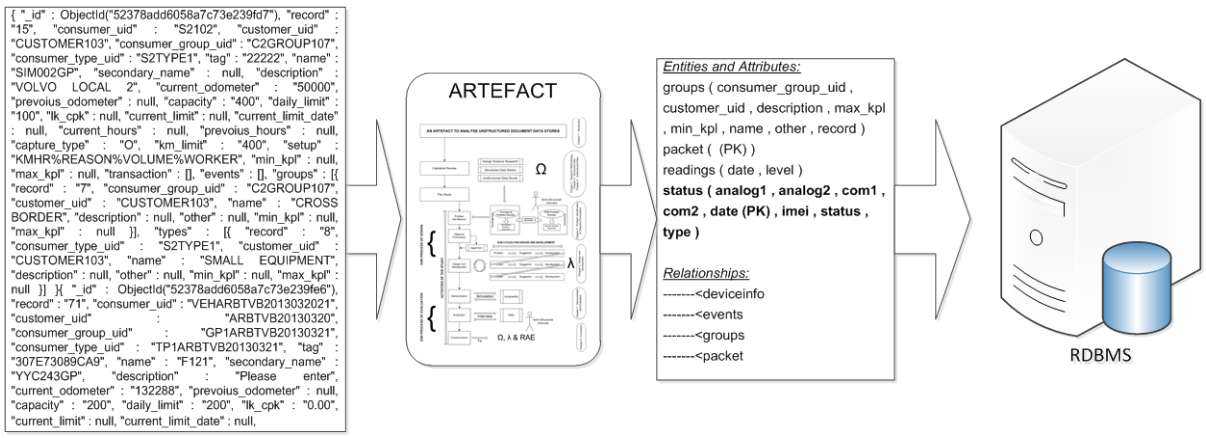


Figure 8.2: Processing objective of data by the artefact (modified)

The formal testing and evaluation of the new object presents an opportunity for further research. This finding supports an empirical objective: to support contributions that could advance the development of further artefacts and technologies that fall under the DSR paradigm.

8.2.3 Conclusions on findings

On self-reflection, this study presented two important determinations:

1. A problem domain and DSR study does not have to start in the RAE.

2. Evaluation of an artefact may lead to new foreseen or unforeseen development cycles, which are beyond the scope of the initial study. It is the responsibility of the researcher to take the initiative on when to conclude the study in terms of reaching objectives.

8.3 RECOMMENDATIONS FOR FUTURE RESEARCH

Various research possibilities were identified during the course of this DSR study. These possibilities present improvements that may be made on the artefact or by expanding the context of the artefact in the RAE. These possibilities include the following:

- The artefact is limited to the analysis of document data stores; expanding the analysis to other data models is a research possibility for the artefact and a new DSR study.
- The artefact's analysis of the data is effective with small data sets in relation to time, but becomes time-consuming with massive data sets. Expanding the artefact to incorporate horizontal scalability and possibly implementing MapReduce, may assist in faster analysis of the data.
- The artefact is limited to only presenting the entities, attributes and relationships; finding a better method of displaying actual data values could provide more information about the data.
- The artefact is limited to one-to-many (1:M) relationships only; future research is required to address many-to-many (M:N) relationships effectively.
- With further regard to the limitations of the artefact: a research opportunity would be to expand the artefact to actually and automatically transform the structure and data into a current RDBMS or NewSQL DBMS.
- Modifying the artefact to incorporate Graphical User Interfaces (GUI) for interaction and visual representation of the data.

- Finally, the actual implementation of the artefact at NFM for thorough evaluation.

8.4 CLOSURE OF THE STUDY

The aim of this study was to design and develop an artefact that could analyse NoSQL document data stores to present acceptable and usable output for a user, such as NFM. This was completed by reviewing the existing literature to investigate descriptive and prescriptive knowledge. The literature and the RAE were investigated to formulate problem domains. The actual design and evaluation of the artefact were done, and this was documented to contribute towards the knowledge base.

This study found that, based on the results presented in Chapter 7, NoSQL document data stores can be analysed using relational data model constructs. This analysis of NoSQL document data stores presents a simplified relational view of the data with the use of an artefact.

However, an unexpected result of the evaluation of the artefact in the RAE led to a design and development cycle of transferring the data to a RDBMS for actual implementation and use. Therefore NoSQL document data stores, commonly found on the web and more specifically the in Web 2.0 domain, may be transferred to some extent to relational database data. This transfer may be done by using and adapting the algorithms presented in Chapters 6 and 7. By applying existing knowledge of structured data to the new field of unstructured data, this study contributed knowledge within the boundaries of the exaptation quadrant of the DSR knowledge contribution framework (KCF) model presented in Figure 2.3 in Chapter 2.

REFERENCE LIST

Adebesin, F., Kotzé, P. & Gelderblom, H. 2011. Design research as a framework to evaluate the usability and accessibility of the digital doorway. (*In* Appiah, E., Mlitwa N. & Anyomi D., *eds*. Proceedings of the 2011 Design, Development and Reseach Conference organised by Cape Peninsula University of Technology. p. 310-327).

Agrawal, G. 2012. What is RDBMS? http://planetofcoders.com/rdbms/ Date of access: 20 June 2013.

Allavena, A., Demers, A. & Hopcroft, J.E. 2005. Correctness of a gossip-based membership protocol. (*In* Proceedings of the twenty-fourth annual ACM Symposium on Principles of Distributed Computing. New York: NY: ACM. p. 292-301).

Anon. 2013. JSON Data Set Sample. http://adobe.github.io/Spry/samples/data_region/JSONDataSetSample.html Date of access: 5 July 2013.

Aslett, M. 2008. Is H-Store the future of database management systems? http://blogs.the451group.com/information_management/2008/03/04/is-h-store-the-future-of-database-management-systems/ Date of access: 11 October 2013.

Aslett, M. 2011. How will the database incumbents respond to NoSQL and NewSQL. *The San Francisco,* 451:1-5.

Attaway, S. 2009. MATLAB: A Practical Introduction to Programming and Problem Solving. Burlingtong: Elsevier Science.

Bakhshi, R., Cloth, L., Fokkink, W. & Haverkort, B. 2009. Mean-field analysis for the evaluation of gossip protocols. (In Sixth International Conference on the Quantitative Evaluation of Systems, 2009 organised by the IEEE. New Brunswick, NJ:IEEE. p. 247-256).

Basho. 2013. Riak. http://basho.com/riak/ Date of access: 5 October 2013

Baskerville, R. 2008. What design science is not. *European Journal of Information Systems,* 17 (5):441-443.

Bentley, L.D. & Whitten, J.L. 2007. Systems Analysis and Design for the Global Enterprise. New York: McGraw-Hill Education.

Bernstein, P.A. & Goodman, N. 1983. Multiversion concurrency control - theory and algorithms. *ACM Transactions on Computer Systems,* 8(4):465-483.

Blanche, M.J.T., Blanche, M.T., Durrheim, K. & Painter, D. 2006. Research in Practice: Applied Methods for the Social Sciences. 3rd ed. Cape Town: University of Cape Town Press.

Bogdan, R.C. & Biklen, S.K. 1982. Qualitative research for education: an introduction to theory and methods. Boston: Allyn & Bacon, Incorporated.

Bondi, A.B. 2000. Characteristics of scalability and their impact on performance. (*In* Proceedings of the 2nd International Workshop on Software and Performance organised by ACM. New York: ACM. p. 195-203).

Borkar, V.R., Carey, M.J. & Li, C. 2012. Big data platforms: What's next? *XRDS: Crossroads, The ACM Magazine for Students,* 19(1):44-49.

Brewer, E.A. 2000. Towards robust distributed systems. (*In* Proceedings of the Annual ACM Symposium on Principles of Distributed Computing organised by ACM. New York: NY: ACM. p.7).

Carvin, A. 2005. Tim Berners-Lee: Weaving a Semantic Web. http://www.andycarvin.com/?p=403 Date of access: 11 October 2013.

Cattell, R. 2011. Scalable SQL and NoSQL data stores. ACM *SIGMOD Record,* 39(4):12-27.

Cervinschi, C.L. & Butucea, D. 2010. Integration of web technologies in software applications. Is Web 2.0 a solution? *Database Systems Journal,* 1(2):39-44.

Chang, F., Dean, J., Ghemawat, S., Hsieh, W.C., Wallach, D.A., Burrows, M., *et al*. 2008. Bigtable: A distributed storage system for structured data. *ACM Transactions on Computer Systems,* 26(2):4.

Codd, E.F. 1970. A relational model of data for large shared data banks. *Communications of the ACM,* 13(6):377-387.

Codd, E.F. 1982. Relational database: a practical foundation for productivity. *Communications of the ACM,* 25(2):109-117.

Codd, E.F. 1985a. Does your DBMS run by the rules? *Computer World,* 21.

Codd, E.F. 1985b. Is your DBMS really relational? *Computer World,* 14.

Connolly, S. 2012. 7 Key Drivers for the Big Data Market. http://hortonworks.com/blog/7-key-drivers-for-the-big-data-market/ Date of access: 10 October 2013.

Connolly, T.M. & Begg, C.E. 2005. Database Systems: A Practical Approach to Design, Implementation, and Management. 4th ed. Harlow: Addison-Wesley.

Connolly, T.M. & Begg, C.E. 2010. Database Systems: A Practical Approach to Design, Implementation, and Management. 5th ed. Harlow: Addison-Wesley.

Connolly, T.M., Begg, C.E. & Holowczak, R. 2008. Business Database Systems. 1st ed. Harlow: Addison-Wesley.

Cormen, T.H., Leiserson, C.E., Rivest, R.L. & Stein, C. 2001. Introduction To Algorithms. Cambridge: MIT Press.

Coronel, C., Morris, S., Rob, P. & Crockett, K. 2014. Database Principles: Fundamentals of Design, Implementation, and Management. 2nd ed. Hampshire: Cengage Learning EMEA.

Creswell, J.W. 2003. Research Design: Qualitative, Quantitative, and Mixed Methods Approaches. 2nd ed. Thousand Oaks: Sage Publications.

Crockford, D. 2006. JSON: The fat-free alternative to XML http://www.json.org/xml.html Date of access: 12 September 2013.

Crockford, D. 2009. Introducing JSON. http://www.json.org Date of access: 3 June 2013.

Dean, J. & Ghemawat, S. 2008. MapReduce: simplified data processing on large clusters. *Communications of the ACM,* 51(1):137-149.

Deering, S. 2011. Example JSON Files, *JQUERY4U AND JSON*. http://www.jquery4u.com/json/10-example-json-files/ Date of access: 3 June 2013.

Deitel, P.J. & Deitel, H.M. 2013. Visual C# 2012: How to Program. 5th ed. Upper Saddle River: Pearson Education, Limited.

Deitel, H.M. & Deitel, P.J. 2006. Visual Basic 2005: How to Program. 3rd ed. Upper Saddle River: Pearson Education, Limited.

DeWitt, D. & Gray, J. 1992. Parallel database systems: the future of high performance database systems. *Communications of the ACM,* 35(6):85-98.

DiNucci, D. 1999. Fragmented Future-Web development faces a process of mitosis, mutation, and natural selection. *PRINT-NEW-YORK,* 53(4):32-35.

Edlich, S. 2009. NoSQL. http://nosql-database.org/index.html Date of access: 20 June 2013.

Evans, E. 2009. NoSQL: What's in a name? http://blog.sym-link.com/2009/10/30/nosql_whats_in_a_name.html Date of access: 20 June 2013.

Faller, S. 2009. Multiversion Concurrency Control. http://www.inf.uni-konstanz.de/dbis/teaching/ss09/tx/Sebastian.pdf Date of access: 20 June 2013.

Gilbert, S. & Lynch, N. 2002. Brewer's conjecture and the feasibility of consistent, available, partition-tolerant web services. *ACM SIGACT News,* 33(2):51-59.

Gopalkrishnan, V., Steier, D., Lewis, H. & Guszcza, J. 2012. Big data, big business: bridging the gap. (*In* Paper presented at the Proceedings of the 1st International Workshop on Big Data, Streams and Heterogeneous Source Mining: Algorithms, Systems, Programming Models and Applications organised by ACM. New York: NY: ACM. p.7-11).

Gregor, S. 2006. The nature of theory in information systems. *MIS Quarterly,* 30(3):611-642.

Gregor, S. & Hevner, A.R. 2013. Positioning and presenting design science research for maximum impact. *MIS Quaterly,* 37(2):337-355.

Gregor, S. & Jones, D. 2007. The Anatomy of a Design Theory. *Journal of the Association for Information Systems,* 8(5):312-335.

Han, J., Haihong, E., Le, G. & Du, J. 2011. Survey on NoSQL database. (*In* 2011 6th International Conference on Pervasive Computing and Applications (ICPCA) organised by IEEE. New Brunswick, NJ:IEEE. p. 336-341).

Hecht, R. & Jablonski, S. 2011. NoSQL evaluation: A use case oriented survey. (*In* 2011 International Conference on Cloud and Service Computing (CSC) organised by IEEE. New Brunswick, NJ:IEEE. p. 336-341).

Helland, P. 2011. If you have too much data, then 'good enough' is good enough. *Communications of the ACM,* 54(6):40-47.

Hevner, A.R. & Chatterjee, S. 2010. Design research in information systems. Vol. 22. New York: Springer.

Hevner, A.R., March, S.T., Park, J. & Ram, S. 2004. Design Science In Information Systems Research. *MIS Quaterly,* 28(1):75-105.

Hoepfl, M.C. 1997. Choosing qualitative research: A primer for technology education researchers. *Journal of Technology Education*, 9(1). http://scholar.lib.vt.edu/ejournals/JTE/v9n1/hoepfl.html Date of access: 20 June 2013.

Hoffer, J.A., Prescott, M.B. & McFadden, F.R. 2007. Modern Database Management. 8th ed. Upper Saddle River: Pearson Education.

Holscher, E., Leifer, C. & Grace, B. 2013. Entity Attribute Value Quickstart. http://akiban.readthedocs.org/en/latest/quickstart/eav.html Date of access: 28 October 2013.

Iivari, J. 2007. A paradigmatic analysis of information systems as a design science. *Scandinavian Journal of Information Systems,* 19(2):39.

Iivari, J. & Venable, J.R. 2009. Action research and design science research - seemingly similar but decisively dissimilar. (*In ECIS 2009 Proceedings*) http://aisel.aisnet.org/ecis2009/73 Date of access: 20 June 2013.

Inmon, W.H. 2005. Building the data warehouse. 3rd ed. New York: Wiley.

Kahn, R.L. & Cannell, C.F. 1957. The dynamics of interviewing; theory, technique, and cases. 4th ed. Oxford: Wiley.

Khan, M.E. 2010. Different Forms of Software Testing Techniques for Finding Errors. *International Journal of Computer Science Issues*, 7 (3):11.

Kimball, R. & Ross, M. 2011. the Data warehouse toolkit: The complete guide to dimensional modeling. 2nd ed. New York: Wiley.

Kimball, R., Ross, M., Thorthwaite, W., Becker, B. & Mundy, J. 2008. The data warehouse lifecycle toolkit. 2nd ed. New York:Wiley.

Leach, P., Mealling, M. & Salz, R. 2005. A Universally Unique IDentifier (UUID) URN Namespace. http://www.ietf.org/rfc/rfc4122.txt Date of access: 17 November 2013.

Leavitt, N. 2010. Will NoSQL Databases Live Up to Their Promise? *Computer,* 43(2):12-14.

Lewin, K. 1946. Action research and minority problems. *Journal of social issues,* 2(4):34-46.

Lith, A. & Mattsson, J. 2010. Investigating storage solutions for large data. Göteborg: Chalmers University of Technology. (Thesis - MSc).

Lohr, S. 2012. The age of big data. *New York Times,* 11.

March, S.T. & Hevner, A.R. 2007. Integrated decision support systems: A data warehousing perspective. *Decision Support Systems,* 43(3):1031-1043.

Marczyk, G.R., DeMatteo, D. & Festinger, D. 2005. Essentials of Research Design and Methodology. 1st ed. Hoboken: Wiley.

Markus, M.L., Majchrzak, A. & Gasser, L. 2002. A design theory for systems that support emergent knowledge processes. *MIS Quarterly,* 26(3):179-212.

Matei, G. 2010. Column-Oriented Databases, an Alternative for Analytical Environment. *Database Systems Journal,* 1 (2):3-16.

Membrey, P., Plugge, E. & Hawkins, T. 2010. The Definitive Guide to MongoDB: The NoSQL Database for Cloud and Desktop Computing. 1st ed. New York: Apress.

Monash, C. 2008. H-Store: Complete destruction of the old DBMS order? http://www.zdnet.com/blog/btl/h-store-complete-destruction-of-the-old-dbms-order/8055 Date of access: 11 October 2013.

MongoDB. 2007. mongoDB. http://www.mongodb.org/ Date of access: 3 July 2013.

MongoDB. 2009. The AGPL, *mongoDB*. http://blog.mongodb.org/post/103832439/the-agpl Date of access: 3 July 2013.

Moniruzzaman, A.B.M. & Hossain, S.A. 2013. NoSQL Database: New Era of Databases for Big data Analytics - Classification, Characteristics and Comparison. *International Journal of Database Theory and Application,* 6(14):1-14.

Morris, S.A., Coronel, C. & Rob, P. 2013. Database Principles: Fundamentals of Design, Implementation, and Management. 10th ed. Cengage Learning.

Myers, M.D. 1997. Qualitative research in information systems. *MIS Quarterly,* 21(2):241-242.

Noor, K.B.M. 2008. Case study: a strategic research methodology. *American Journal of Applied Sciences,* 5(11):1602-1604.

North, K. 2009. Databases in the Cloud: Elysian Fields or Briar Patch? http://www.drdobbs.com/database/databases-in-the-cloud-elysian-fields-or/218900502 Date of access: 3 July 2013.

Oates, B.J. 2006. Researching Information Systems and Computing. Los Angeles: SAGE Publications.

Okman, L., Gal-Oz, N., Gonen, Y., Gudes, E. & Abramov, J. 2011. Security issues in nosql databases. (*In* 2011 IEEE 10th International Conference on Trust, Security and Privacy in Computing and Communications (TrustCom) organised by IEEE. New Brunswick, NJ:IEEE. p. 541-547).

Oliveira da Silva, C.A.R.F. 2011. Data Modeling with NoSQL: How, When and Why. Porto:University of Porto. (Dissertation - MSc).

Olson, M.A., Bostic, K. & Seltzer, M.I. 1999. Berkeley DB. (*In* USENIX Annual Technical Conference, FREENIX Track organised by USENIX. p. 183-191).

Oosthuizen, F. 2013a. General Disscustion [personal interview]. 1 Jul., Vanderbijlpark.

Oosthuizen, F. 2013b. Situation Analysis [personal interview]. 1 Oct., Vanderbijlpark.

Oosthuizen, F. 2013c. Evaluation Analysis [personal interview]. 6 Nov., Vanderbijlpark.

O'Reilly, T. 2005. Web 2.0: Compact Definition? http://radar.oreilly.com/2005/10/web-20-compact-definition.html Date of access: 11 October 2013.

O'Reilly, T. 2006. Web 2.0 Compact Definition: Trying Again. http://radar.oreilly.com/2006/12/web-20-compact-definition-tryi.html Date of access: 10 October 2013.

Oxford Dictionaries & Press. 2013. Pocket Oxford English Dictionary [CD].

Padhy, R.P., Patra, M.R. & Satapathy, S.C. 2011. RDBMS to NoSQL: Reviewing Some Next-Generation Non-Relational Database‘s. *International Journal of Advanced Engineering Science and Technologies,* 11(1):15-30.

Peffers, K., Tuunanen, T., Rothenberger, M.A. & Chatterjee, S. 2008. A Design Science Research Methodology for Information Systems Research. *Journal of Management Information Systems,* 24(3):45-77.

Pokorny, J. 2011. NoSQL databases: a step to database scalability in web environment. (*In* Proceedings of the 13th International Conference on Information Integration and Web-based Applications and Services organised by ACM. New York: NY: ACM. p.278-283).

Pritchett, D. 2008. Base: an ACID alternative. *Queue,* 6(3):48-55.

Purao, S. 2002. Design research in the technology of information systems: Truth or dare. *http://purao.ist.psu.edu/working-papers/dare-purao.pdf* Date of access: 15 July 2013.

Rahul, R. 2008. Shard – A Database Design. http://technoroy.blogspot.com/2008/07/shard-database-design.html Date of access: 11 October 2013.

Ramakrishnan, R. & Gehrke, J. 2003. Database Management Systems. 3rd ed. Michigan: McGraw-Hill Education.

Rapoport, R.N. 1970. Three Dilemmas in Action Research: With Special Reference to the Tavistock Experience. *Human Relations,* 23(6):499-513.

Reed, D.P. 1978. Naming and synchronization in a decentralized computer system. http://publications.csail.mit.edu/lcs/pubs/pdf/MIT-LCS-TR-205.pdf Date of access: 3 July 2013.

Remenyi, D., Williams, B., Money, A. & Swartz, E. 1998. Doing research in business and management: an introduction to process and method. 1st ed. London: Sage Publications Limited.

Richardson, W. 2009. Blogs, Wikis, Podcasts, and Other Powerful Web Tools for Classrooms. 2nd ed. Thousand Oaks: SAGE Publications.

Ries, E. 2009. Sharding for startups. http://www.startuplessonslearned.com/2009/01/sharding-for-startups.html Date of access: 11 October 2013.

Rob, P., Coronel, C. & Crockett, K. 2008. Database systems: design, implementation & management. London: Cengage Learning.

Robinson, I., Webber, J. & Eifrem, E. 2013. Graph Databases. 1st ed. Sebastopol: O'Reilly Media, Inc.

Rogers, Y., Sharp, H. & Preece, J. 2011. Interaction Design: Beyond Human - Computer Interaction. New York:Wiley.

Rud, O.P. 2009. Business Intelligence Success Factors: Tools for Aligning Your Business in the Global Economy. Hoboken: Wiley.

Saunders, M., Lewis, P. & Thornhill, A. 2009. Research methods for business students. 5th ed. Harlow: Pearson.

Seaman, C.B. 1999. Qualitative methods in empirical studies of software engineering. *IEEE Transactions on Software Engineering,* 25(4):557-572.

Shore, J. 2004. Continuous design. *Software, IEEE,* 21(1):20-22.

Simon, H.A. 1996. The sciences of the artificial. 3rd ed. Cambridge: MIT Press.

Stokes, D.E. 1997. Pasteur's quadrant: Basic science and technological innovation. Washington: Brookings Institution Press.

Stonebraker, M. 1986. The case for shared nothing. *IEEE Database Engineering Bulletin,* 9 (1):4-9.

Stonebraker, M. 2010a. In search of database consistency. *Communications of the ACM,* 53(10):8-9.

Stonebraker, M. 2010b. SQL databases vs. NoSQL databases. *Communications of the ACM,* 53(4):10-11.

Straub, D.W., Ang, S. & Evaristo, R. 1994. Normative standards for IS research. *SIGMIS Database,* 25(1):21-34.

Strauch, C. & Kriha, W. 2011. *NoSQL databases*. http://www.christof-strauch.de/nosqldbs.pdf Date of access: 3 July 2013.

Strauss, A. & Corbin, J.M. 1990. Basics of Qualitative Research: Techniques and Procedures for Developing Grounded Theory. 17th ed. Michigan: SAGE Publications.

The-451-Group. 2013. NoSQL LinkedIn Skills Index. http://blogs.the451group.com/information_management/?s=NoSQL+LinkedIn+Skills+Index Date of access: 24 September 2013.

Tiwari, S. 2011. Professional NoSQL. Indianapolis: Wiley.

Vaishnavi, V. & Kuechler, K. 2004. Design Research in Information Systems. http://desrist.org/design-research-in-information-systems/ Date of access: 14 June 2013.

Vogels, W. 2009. Eventually consistent. *Communications of the ACM,* 52(1):40-44.

Walls, J.G., Widmeyer, G.R. & El Sawy, O.A. 1992. Building an information system design theory for vigilant EIS. *Information systems research,* 3(1):36-59.

Wang, D., Chen, R. & Chu, C.-P. 2000. Analyzing reconfigurable algorithms for managing replicated data with strict consistency requirements: a case study. (*In* The 24th Annual International Conference on Computer Software and Applications Conference, 2000 organised by IEEE. New Brunswick, NJ:IEEE. p. 608-613)

Weber, S. 2011? NoSQL Databases. http://wiki.hsr.ch/Datenbanken/files/Weber_NoSQL_Paper.pdf Date of access: 10 September 2011

APPENDIX A: RAE SAMPLE DATA

(Values have been replaced to maintain privacy)

{ "_id" : ObjectId("52378add6058a7c73e239fd7"), "record" : "15", "consumer_uid" : "S2102", "customer_uid" : "CUSTOMER103", "consumer_group_uid" : "C2GROUP107", "consumer_type_uid" : "S2TYPE1", "tag" : "22222", "name" : "SIM002GP", "secondary_name" : null, "description" : "VOLVO LOCAL 2", "current_odometer" : "50000", "prevoius_odometer" : null, "capacity" : "400", "daily_limit" : "100", "lk_cpk" : null, "current_limit" : null, "current_limit_date" : null, "current_hours" : null, "prevoius_hours" : null, "capture_type" : "O", "km_limit" : "400", "setup" : "KMHR%REASON%VOLUME%WORKER", "min_kpl" : null, "max_kpl" : null, "transaction" : [], "events" : [], "groups" : [{ "record" : "7", "consumer_group_uid" : "C2GROUP107", "customer_uid" : "CUSTOMER103", "name" : "CROSS BORDER", "description" : null, "other" : null, "min_kpl" : null, "max_kpl" : null }], "types" : [{ "record" : "8", "consumer_type_uid" : "S2TYPE1", "customer_uid" : "CUSTOMER103", "name" : "SMALL EQUIPMENT", "description" : null, "other" : null, "min_kpl" : null, "max_kpl" : null }] }{ "_id" : ObjectId("52378add6058a7c73e239fe6"), "record" : "71", "consumer_uid" : "VEHARBTVB2013032021", "customer_uid" : "ARBTVB20130320", "consumer_group_uid" : "GP1ARBTVB20130321", "consumer_type_uid" : "TP1ARBTVB20130321", "tag" : "307E73089CA9", "name" : "F121", "secondary_name" : "YYC243GP", "description" : "Please enter", "current_odometer" : "132288", "prevoius_odometer" : null, "capacity" : "200", "daily_limit" : "200", "lk_cpk" : "0.00", "current_limit" : null, "current_limit_date" : null, "current_hours" : null, "prevoius_hours" : null, "capture_type" : "O", "km_limit" : "500", "setup" : "KMHR%REASON%VOLUME%WORKER", "min_kpl" : "7", "max_kpl" : "11", "transaction" : [{ "record" : "335", "consumer_transaction_uid" : "48da8547-a818-11e2-b3d0-001c14012e75", "udatetime" : "2013-04-18 01:02:25", "consumer_uid" : "VEHARBTVB2013032021", "worker_uid" : "OP1ARBTVB201303207", "volume" : "0.061", "odometer" : "132302.000", "hours" : "0", "source" : "Internal", "location" : "Unknown", "price" : "0.06", "remarks" : "Other Filling", "start_time" : "2013-04-18 01:01:57", "end_time" : "2013-04-18 01:02:32", "cpk" : "0.00", "kpl" : "0.00", "bypassed" : "NO", "auth_uid" : "Unknown", "restype" : "Normal", "trip_distance" : "0" }, { "record" : "425", "consumer_transaction_uid" : "8a2f250b-ae40-11e2-b3d0-001c14012e75", "udatetime" : "2013-04-26 09:08:17", "consumer_uid" : "VEHARBTVB2013032021", "worker_uid" : "OP1ARBTVB201303223", "volume" : "49.413", "odometer" : "135125.000", "hours" : "0", "source" : "Internal", "location" : "Unknown", "price" : "49.41", "remarks" : "Normal Filling", "start_time" : "2013-04-26 09:07:39", "end_time" : "2013-04-26 09:09:40", "cpk" : "57.13", "kpl" : "57.13", "bypassed" : "NO", "auth_uid" : "Unknown", "restype" : "Normal", "trip_distance" : "2823" }, { "record" : "467", "consumer_transaction_uid" : "6116f40d-b3c1-11e2-b3d0-001c14012e75", "udatetime" : "2013-05-03 09:04:22", "consumer_uid" : "VEHARBTVB2013032021", "worker_uid" : "OP1ARBTVB201303221", "volume" : "48.900", "odometer" : "135524.000", "hours" : "0", "source" : "Internal", "location" : "Unknown", "price" : "48.90", "remarks" : "Normal Filling", "start_time" : "2013-05-03 09:03:04", "end_time" : "2013-05-03 09:05:48", "cpk" : "8.16", "kpl" : "8.16", "bypassed" : "NO", "auth_uid" : "Unknown", "restype" : "Normal", "trip_distance" : "399" }], "events" : [{ "record" : "289", "data_event_uid" : "1b85b70e-ddd1-4e74-9dc6-eb2015ac36fc", "udatetime" : "2013-05-19 11:55:04", "customer_uid" : "0", "consumer_uid" : "VEHARBTVB2013032021", "depot_uid" : "NA", "station_uid" : "NA", "container_uid" : "NA", "worker_uid" : "0", "event_name" : "Consumer F121 KPL below Minimum", "event_type" : "Consumer KPL Below Minimum", "description" : "Consumer F121 KPL of 0.000000 is below the minimum of 7.", "processed" : "1" }, { "record" : "535", "data_event_uid" : "52153f07-d291-44e8-b3ff-0d1404f13f25", "udatetime" : "2013-06-02 11:55:00", "customer_uid" : "ARBTVB20130320", "consumer_uid" : "VEHARBTVB2013032021", "depot_uid" : "NA", "station_uid" : "NA", "container_uid" : "NA", "worker_uid" : "0", "event_name" : "Consumer F121 KPL below Minimum", "event_type" : "Consumer KPL Below Minimum", "description" : "Consumer F121 KPL of 0.000000 is below the minimum of 7.", "processed" : "1" }], "groups" : [{ "record" : "10", "consumer_group_uid" : "GP1ARBTVB20130321", "customer_uid" : "ARBTVB20130320", "name" : "Bakkies", "description" : null, "other" : null, "min_kpl" : null, "max_kpl" : null }], "types" : [{ "record" : "12", "consumer_type_uid" : "TP1ARBTVB20130321", "customer_uid" : "ARBTVB20130320", "name" : "Bakkies", "description" : null, "other" : null, "min_kpl" : null, "max_kpl" : null }] }{ "_id" : ObjectId("52378add6058a7c73e239fec"), "record" : "77", "consumer_uid" : "VEHARBTVB2013032027", "customer_uid" : "ARBTVB20130320", "consumer_group_uid" : "GP1ARBTVB20130321", "consumer_type_uid" : "TP1ARBTVB20130321", "tag" : "307E72E0A37F", "name" : "F131", "secondary_name" : "BY78XSGP", "description" : "Please enter", "current_odometer" : "121503", "prevoius_odometer" : null, "capacity" : "200", "daily_limit" : "200", "lk_cpk" : "0.00", "current_limit" : null, "current_limit_date" : null, "current_hours" : null, "prevoius_hours" : null, "capture_type" : "O", "km_limit" : "500", "setup" : "KMHR%REASON%VOLUME%WORKER", "min_kpl" : "0", "max_kpl" : "0", "transaction" : [{ "record" : "297", "consumer_transaction_uid" : "1a38b7bb-8573-11e2-8602-000c29513585", "udatetime" : "2013-03-20 09:24:22", "consumer_uid" : "VEHARBTVB2013032027", "worker_uid" : "OP1ARBTVB201303201", "volume" : "180.000", "odometer" : "121506.000", "hours" : "0", "source" : "Internal", "location" : null, "price" : "180.00", "remarks" : "REASON 2", "start_time" : "2013-03-20 09:23:45", "end_time" : "2013-03-20 09:25:02", "cpk" : "0.01", "kpl" : "0.01", "bypassed" : null, "auth_uid" : null, "restype" : "diesel", "trip_distance" : "2" }, { "record" : "298", "consumer_transaction_uid" : "1a3bbc0c-8573-11e2-8602-000c29513585", "udatetime" : "2013-03-20 09:23:29", "consumer_uid" : "VEHARBTVB2013032027", "worker_uid" : "OP1ARBTVB201303201", "volume" : "22.000", "odometer" : "121504.000", "hours" : "0", "source" : "Internal", "location" : null, "price" : "22.00", "remarks" : "REASON 2", "start_time" : "2013-03-20 09:23:03", "end_time" : "2013-03-20 09:23:39", "cpk" : null, "kpl" : null, "bypassed" : null, "auth_uid" : null, "restype" : "diesel", "trip_distance" : "0" }, { "record" : "754", "consumer_transaction_uid" : "abf885b2-d4f0-11e2-b3d0-001c14012e75", "udatetime" : "2013-06-14 02:17:40", "consumer_uid" : "VEHARBTVB2013032027", "worker_uid" : "OP1ARBTVB201303229", "volume" : "34.356", "odometer" : "122066.000", "hours" : "0", "source" : "Internal", "location" : "Unknown", "price" : "34.36", "remarks" : "Normal Filling", "start_time" : "2013-06-14 02:17:17", "end_time" : "2013-06-14 02:18:37", "cpk" : "16.30", "kpl" : "16.30", "bypassed" : "NO", "auth_uid" : "Unknown", "restype" : "Normal", "trip_distance" : "560" }], "events" : [{ "record" : "1351", "data_event_uid" : "cc44d2b8-7fbd-45f0-94d1-d60c3ec3d2f1", "udatetime" : "2013-06-16 11:55:01", "customer_uid" : "ARBTVB20130320", "consumer_uid" : "VEHARBTVB2013032027", "depot_uid" : "NA", "station_uid" : "NA", "container_uid" : "NA", "worker_uid" : "0", "event_name" : "Consumer F131 KPL above Maximum", "event_type" : "Consumer KPL Above Maximum", "description" : "Consumer F131 KPL of 16.300000 is above the maximum of 0.", "processed" : "1" }], "groups" : [{ "record" : "10", "consumer_group_uid" : "GP1ARBTVB20130321", "customer_uid" : "ARBTVB20130320", "name" : "Bakkies", "description" : null, "other" : null, "min_kpl" : null, "max_kpl" : null }], "types" : [{ "record" : "12", "consumer_type_uid" : "TP1ARBTVB20130321", "customer_uid" : "ARBTVB20130320", "name" : "Bakkies", "description" : null, "other" : null, "min_kpl" : null, "max_kpl" : null }] }{ "_id" : ObjectId("52378add6058a7c73e239ff1"), "record" : "82", "consumer_uid" : "VEHARBTVB2013032032", "customer_uid" : "ARBTVB20130320", "consumer_group_uid" : "GP1ARBTVB20130321", "consumer_type_uid" : "TP1ARBTVB20130321", "tag" : "307E7306506B", "name" : "F120", "secondary_name" : "YYC233GP", "description" : "Please enter", "current_odometer" : "148116", "prevoius_odometer" : null, "capacity" : "200", "daily_limit" : "200", "lk_cpk" : "0.00", "current_limit" : null, "current_limit_date" : null, "current_hours" : null, "prevoius_hours" : null, "capture_type" : "O", "km_limit" : "500", "setup" : "KMHR%REASON%VOLUME%WORKER", "min_kpl" : "7", "max_kpl" : "11", "transaction" : [{ "record" : "401", "consumer_transaction_uid" : "07e8afd1-aca3-11e2-b3d0-001c14012e75", "udatetime" : "2013-04-24 07:47:29", "consumer_uid" : "VEHARBTVB2013032032", "worker_uid" : "OP1ARBTVB201303223", "volume" : "50.978", "odometer" : "150000.000", "hours" : "0", "source" : "Internal", "location" : "Unknown", "price" : "50.98", "remarks" : "Normal Filling", "start_time" : "2013-04-24 07:46:32", "end_time" : "2013-04-24 07:49:31", "cpk" : "271.93", "kpl" : "271.94", "bypassed" : "NO", "auth_uid" : "Unknown", "restype" : "Normal", "trip_distance" : "13863" }, { "record" : "439", "consumer_transaction_uid" : "762c543a-b096-11e2-b3d0-001c14012e75", "udatetime" : "2013-04-29 08:26:38", "consumer_uid" : "VEHARBTVB2013032032", "worker_uid" : "OP1ARBTVB201303222", "volume" : "30.797", "odometer" : "150441.000", "hours" : "0", "source" : "Internal", "location" : "Unknown", "price" : "30.80", "remarks" : "Normal Filling", "start_time" : "2013-04-29 08:25:28", "end_time" : "2013-04-29 08:28:14", "cpk" : "0.06", "kpl" : "0.06", "bypassed" : "NO", "auth_uid" : "Unknown", "restype" : "Normal", "trip_distance" : "2" }, { "record" : "440", "consumer_transaction_uid" : "765992fb-b096-11e2-b3d0-001c14012e75", "udatetime" : "2013-04-29 08:24:28", "consumer_uid" : "VEHARBTVB2013032032", "worker_uid" : "OP1ARBTVB201303222", "volume" : "20.250", "odometer" : "150439.000", "hours" : "0", "source" : "Internal", "location" : "Unknown", "price" : "20.25", "remarks" : "Normal Filling", "start_time" : "2013-04-29 08:23:31", "end_time" : "2013-04-29 08:24:59", "cpk" : "7429.04", "kpl" : "7429.04", "bypassed" : "NO", "auth_uid" : "Unknown", "restype" : "Normal", "trip_distance" : "150438" }], "events" : [{ "record" : "1062", "data_event_uid" : "8b3809e8-604d-4aa9-ab65-95378b7e4f86", "udatetime" : "2013-06-09 11:55:01", "customer_uid" : "ARBTVB20130320", "consumer_uid" : "VEHARBTVB2013032032", "depot_uid" : "NA", "station_uid" : "NA", "container_uid" : "NA", "worker_uid" : "0", "event_name" : "Consumer F120

KPL above Maximum", "event_type" : "Consumer KPL Above Maximum", "description" : "Consumer F120 KPL of 14.130000 is above the maximum of 11.", "processed" : "1" }, { "record" : "1335", "data_event_uid" : "8b3809e8-604d-4aa9-ab65-95378b7e4f86", "udatetime" : "2013-06-09 11:55:01", "customer_uid" : "ARBTVB20130320", "consumer_uid" : "VEHARBTVB2013032032", "depot_uid" : "NA", "station_uid" : "NA", "container_uid" : "NA", "worker_uid" : "0", "event_name" : "Consumer F120 KPL above Maximum", "event_type" : "Consumer KPL Above Maximum", "description" : "Consumer F120 KPL of 14.130000 is above the maximum of 11.", "processed" : "1" }], "groups" : [{ "record" : "10", "consumer_group_uid" : "GP1ARBTVB20130321", "customer_uid" : "ARBTVB20130320", "name" : "Bakkies", "description" : null, "other" : null, "min_kpl" : null, "max_kpl" : null }], "types" : [{ "record" : "12", "consumer_type_uid" : "TP1ARBTVB20130321", "customer_uid" : "ARBTVB20130320", "name" : "Bakkies", "description" : null, "other" : null, "min_kpl" : null, "max_kpl" : null }] }{ "_id" : ObjectId("52378add6058a7c73e239ff3"), "record" : "84", "consumer_uid" : "VEHARBTVB2013032034", "customer_uid" : "ARBTVB20130320", "consumer_group_uid" : "GP1ARBTVB20130321", "consumer_type_uid" : "TP1ARBTVB20130321", "tag" : "307E729E4FED", "name" : "F136", "secondary_name" : "BD33DMGP", "description" : "Please enter", "current_odometer" : "269718", "prevoius_odometer" : null, "capacity" : "200", "daily_limit" : "200", "lk_cpk" : "0.00", "current_limit" : null, "current_limit_date" : null, "current_hours" : null, "prevoius_hours" : null, "capture_type" : "O", "km_limit" : "500", "setup" : "KMHR%REASON%VOLUME%WORKER", "min_kpl" : "7", "max_kpl" : "11", "transaction" : [{ "record" : "313", "consumer_transaction_uid" : "1e05e42f-a776-11e2-b3d0-001c14012e75", "udatetime" : "2013-04-17 05:40:17", "consumer_uid" : "VEHARBTVB2013032034", "worker_uid" : "OP1ARBTVB201303228", "volume" : "41.679", "odometer" : "269720.000", "hours" : "0", "source" : "Internal", "location" : "Unknown", "price" : "41.68", "remarks" : "Normal Filling", "start_time" : "2013-04-17 05:39:25", "end_time" : "2013-04-17 05:42:57", "cpk" : "5458.21", "kpl" : "5458.34", "bypassed" : "NO", "auth_uid" : "Unknown", "restype" : "Normal", "trip_distance" : "227498" }, { "record" : "314", "consumer_transaction_uid" : "a151175f-a77d-11e2-b3d0-001c14012e75", "udatetime" : "2013-04-17 06:40:17", "consumer_uid" : "VEHARBTVB2013032034", "worker_uid" : "OP1ARBTVB201303228", "volume" : "0.051", "odometer" : "269721.000", "hours" : "0", "source" : "Internal", "location" : "Unknown", "price" : "0.05", "remarks" : "Other Filling", "start_time" : "2013-04-17 06:39:56", "end_time" : "2013-04-17 06:40:23", "cpk" : "20.00", "kpl" : "19.61", "bypassed" : "NO", "auth_uid" : "Unknown", "restype" : "Normal", "trip_distance" : "1" }, { "record" : "323", "consumer_transaction_uid" : "37599cb3-a811-11e2-b3d0-001c14012e75", "udatetime" : "2013-04-18 12:15:58", "consumer_uid" : "VEHARBTVB2013032034", "worker_uid" : "OP1ARBTVB201303215", "volume" : "0.025", "odometer" : "269722.000", "hours" : "0", "source" : "Internal", "location" : "Unknown", "price" : "0.03", "remarks" : "Other Filling", "start_time" : "2013-04-18 12:15:16", "end_time" : "2013-04-18 12:16:08", "cpk" : "33.33", "kpl" : "40.00", "bypassed" : "NO", "auth_uid" : "Unknown", "restype" : "Normal", "trip_distance" : "1" }], "events" : [{ "record" : "295", "data_event_uid" : "ff1815d2-717f-4a0b-8352-c097bfadfd45", "udatetime" : "2013-05-19 11:55:04", "customer_uid" : "ARBTVB20130320", "consumer_uid" : "VEHARBTVB2013032034", "depot_uid" : "NA", "station_uid" : "NA", "container_uid" : "NA", "worker_uid" : "0", "event_name" : "Consumer F136 KPL above Maximum", "event_type" : "Consumer KPL Above Maximum", "description" : "Consumer F136 KPL of 18.710000 is above the maximum of 11.", "processed" : "1" }, { "record" : "377", "data_event_uid" : "a4620829-b17f-46db-bbec-b813fabf755a", "udatetime" : "2013-05-26 11:55:02", "customer_uid" : "ARBTVB20130320", "consumer_uid" : "VEHARBTVB2013032034", "depot_uid" : "NA", "station_uid" : "NA", "container_uid" : "NA", "worker_uid" : "0", "event_name" : "Consumer F136 KPL below Minimum", "event_type" : "Consumer KPL Below Minimum", "description" : "Consumer F136 KPL of 6.980000 is below the minimum of 7.", "processed" : "1" }, { "record" : "541", "data_event_uid" : "9ac14577-6d2a-4fe1-9dcb-8a6802f4697f", "udatetime" : "2013-06-02 11:55:00", "customer_uid" : "ARBTVB20130320", "consumer_uid" : "VEHARBTVB2013032034", "depot_uid" : "NA", "station_uid" : "NA", "container_uid" : "NA", "worker_uid" : "0", "event_name" : "Consumer F136 KPL above Maximum", "event_type" : "Consumer KPL Above Maximum", "description" : "Consumer F136 KPL of 14.816667 is above the maximum of 11.", "processed" : "1" }], "groups" : [{ "record" : "10", "consumer_group_uid" : "GP1ARBTVB20130321", "customer_uid" : "ARBTVB20130320", "name" : "Bakkies", "description" : null, "other" : null, "min_kpl" : null, "max_kpl" : null }], "types" : [{ "record" : "12", "consumer_type_uid" : "TP1ARBTVB20130321", "customer_uid" : "ARBTVB20130320", "name" : "Bakkies", "description" : null, "other" : null, "min_kpl" : null, "max_kpl" : null }] }{ "_id" : ObjectId("52378add6058a7c73e239ff4"), "record" : "85", "consumer_uid" : "VEHARBTVB2013032035", "customer_uid" : "ARBTVB20130320", "consumer_group_uid" : "GP1ARBTVB20130323", "consumer_type_uid" : "TP1ARBTVB20130323", "tag" : "307E7307506A", "name" : "TGV797GP", "secondary_name" : "RENAULT", "description" : "Please enter", "current_odometer" : "120276", "prevoius_odometer" : null, "capacity" : "200", "daily_limit" : "200", "lk_cpk" : "0.00", "current_limit" : null, "current_limit_date" : null, "current_hours" : null, "prevoius_hours" : null, "capture_type" : "O", "km_limit" : "500", "setup" : "KMHR%REASON%VOLUME%WORKER", "min_kpl" : "7", "max_kpl" : "11", "transaction" : [{ "record" : "327", "consumer_transaction_uid" : "b384e62e-a811-11e2-b3d0-001c14012e75", "udatetime" : "2013-04-18 12:20:56", "consumer_uid" : "VEHARBTVB2013032035", "worker_uid" : "OP1ARBTVB201303213", "volume" : "0.005", "odometer" : "120277.000", "hours" : "0", "source" : "Internal", "location" : "Unknown", "price" : "0.01", "remarks" : "Other Filling", "start_time" : "2013-04-18 12:20:27", "end_time" : "2013-04-18 12:21:03", "cpk" : "-100.00", "kpl" : "-200.00", "bypassed" : "NO", "auth_uid" : "Unknown", "restype" : "Normal", "trip_distance" : "-1" }, { "record" : "336", "consumer_transaction_uid" : "a9109157-a820-11e2-b3d0-001c14012e75", "udatetime" : "2013-04-18 02:01:27", "consumer_uid" : "VEHARBTVB2013032035", "worker_uid" : "OP1ARBTVB201303228", "volume" : "60.720", "odometer" : "120278.000", "hours" : "0", "source" : "Internal", "location" : "Unknown", "price" : "60.72", "remarks" : "Normal Filling", "start_time" : "2013-04-18 02:00:51", "end_time" : "2013-04-18 02:03:52", "cpk" : "0.00", "kpl" : "0.00", "bypassed" : "NO", "auth_uid" : "Unknown", "restype" : "Normal", "trip_distance" : "0" }, { "record" : "592", "consumer_transaction_uid" : "a3b6b908-c470-11e2-b3d0-001c14012e75", "udatetime" : "2013-05-22 01:03:56", "consumer_uid" : "VEHARBTVB2013032035", "worker_uid" : "OP1ARBTVB201303230", "volume" : "61.015", "odometer" : "122073.000", "hours" : "0", "source" : "Internal", "location" : "Unknown", "price" : "61.01", "remarks" : "Normal Filling", "start_time" : "2013-05-22 01:03:08", "end_time" : "2013-05-22 01:06:28", "cpk" : "29.44", "kpl" : "29.44", "bypassed" : "NO", "auth_uid" : "Unknown", "restype" : "Normal", "trip_distance" : "1796" }], "events" : [{ "record" : "378", "data_event_uid" : "6b72ed12-a42e-4ee6-b614-7819f4339a43", "udatetime" : "2013-05-26 11:55:02", "customer_uid" : "ARBTVB20130320", "consumer_uid" : "VEHARBTVB2013032035", "depot_uid" : "NA", "station_uid" : "NA", "container_uid" : "NA", "worker_uid" : "0", "event_name" : "Consumer TGV797GP KPL below Minimum", "event_type" : "Consumer KPL Below Minimum", "description" : "Consumer TGV797GP KPL of 0.000000 is below the minimum of 7.", "processed" : "1" }, { "record" : "1915", "data_event_uid" : "17bbd773-156c-4cb7-9699-be2a3e4949e8", "udatetime" : "2013-07-07 11:55:02", "customer_uid" : "ARBTVB20130320", "consumer_uid" : "VEHARBTVB2013032035", "depot_uid" : "NA", "station_uid" : "NA", "container_uid" : "NA", "worker_uid" : "0", "event_name" : "Consumer TGV797GP KPL above Maximum", "event_type" : "Consumer KPL Above Maximum", "description" : "Consumer TGV797GP KPL of 13.710000 is above the maximum of 11.", "processed" : "1" }, { "record" : "1944", "data_event_uid" : "17bbd773-156c-4cb7-9699-be2a3e4949e8", "udatetime" : "2013-07-07 11:55:02", "customer_uid" : "ARBTVB20130320", "consumer_uid" : "VEHARBTVB2013032035", "depot_uid" : "NA", "station_uid" : "NA", "container_uid" : "NA", "worker_uid" : "0", "event_name" : "Consumer TGV797GP KPL above Maximum", "event_type" : "Consumer KPL Above Maximum", "description" : "Consumer TGV797GP KPL of 13.710000 is above the maximum of 11.", "processed" : "1" }], "groups" : [{ "record" : "12", "consumer_group_uid" : "GP1ARBTVB20130323", "customer_uid" : "ARBTVB20130320", "name" : "Private", "description" : null, "other" : null, "min_kpl" : null, "max_kpl" : null }], "types" : [{ "record" : "14", "consumer_type_uid" : "TP1ARBTVB20130323", "customer_uid" : "ARBTVB20130320", "name" : "Private", "description" : null, "other" : null, "min_kpl" : null, "max_kpl" : null }] }{ "_id" : ObjectId("52378add6058a7c73e239ffc"), "record" : "93", "consumer_uid" : "VEHARBTVB2013032043", "customer_uid" : "ARBTVB20130320", "consumer_group_uid" : "GP1ARBTVB20130320", "consumer_type_uid" : "TP1ARBTVB20130320", "tag" : "307E729FE241", "name" : "F110", "secondary_name" : "WWV118GP", "description" : "Please enter", "current_odometer" : "651877", "prevoius_odometer" : null, "capacity" : "1000", "daily_limit" : "1000", "lk_cpk" : "0.00", "current_limit" : null, "current_limit_date" : null, "current_hours" : null, "prevoius_hours" : null, "capture_type" : "O", "km_limit" : "500", "setup" : "KMHR%REASON%VOLUME%WORKER", "min_kpl" : "2.3", "max_kpl" : "2.6", "transaction" : [{ "record" : "392", "consumer_transaction_uid" : "b11de349-ac1e-11e2-b3d0-001c14012e75", "udatetime" : "2013-04-23 03:57:23", "consumer_uid" : "VEHARBTVB2013032043", "worker_uid" : "OP1ARBTVB201303205", "volume" : "255.192", "odometer" : "651878.000", "hours" : "0", "source" : "Internal", "location" : "Unknown", "price" : "255.19", "remarks" : "Normal Filling", "start_time" : "2013-04-23 03:56:04", "end_time" : "2013-04-23 04:02:55", "cpk" : "0.00", "kpl" : "0.00", "bypassed" : "NO", "auth_uid" : "Unknown", "restype" : "Normal", "trip_distance" : "0" }, { "record" : "444", "consumer_transaction_uid" : "0be08ced-b0bc-11e2-b3d0-001c14012e75", "udatetime" : "2013-04-29 12:43:16", "consumer_uid" : "VEHARBTVB2013032043", "worker_uid" : "OP1ARBTVB201303205", "volume" : "542.313", "odometer" : "660816.000", "hours" : "0", "source" : "Internal", "location" : "Unknown", "price" : "542.31", "remarks" : "Normal Filling", "start_time" : "2013-04-29 12:42:24", "end_time" : "2013-04-29 12:58:46", "cpk" : "16.48", "kpl" : "16.48", "bypassed" : "NO", "auth_uid" : "Unknown", "restype" : "Normal", "trip_distance" : "8938" }, { "record" : "455", "consumer_transaction_uid" : "ee3464f0-b20d-11e2-b3d0-001c14012e75", "udatetime" : "2013-04-30 08:28:48", "consumer_uid" : "VEHARBTVB2013032043", "worker_uid" : "OP1ARBTVB201303205", "volume" : "550.809", "odometer" : "662351.000", "hours" : "0", "source" : "Internal", "location" : "Unknown", "price" : "550.81", "remarks" : "Normal Filling", "start_time" : "2013-04-30 08:28:02", "end_time" : "2013-04-30 08:42:15", "cpk" : "2.79", "kpl" : "2.79", "bypassed" : "NO", "auth_uid" :

"Unknown", "restype" : "Normal", "trip_distance" : "1535" }], "events" : [{ "record" : "299", "data_event_uid" : "22eb3dfc-ac89-467d-8d02-5221bdab0e21", "udatetime" : "2013-05-19 11:55:04", "customer_uid" : "0", "consumer_uid" : "VEHARBTVB2013032043", "depot_uid" : "NA", "station_uid" : "NA", "container_uid" : "NA", "worker_uid" : "0", "event_name" : "Consumer F110 KPL below Minimum", "event_type" : "Consumer KPL Below Minimum", "description" : "Consumer F110 KPL of 0.000000 is below the minimum of 2.3.", "processed" : "1" }, { "record" : "383", "data_event_uid" : "50d54c91-3875-4893-b20e-303e264abd97", "udatetime" : "2013-05-26 11:55:02", "customer_uid" : "ARBTVB20130320", "consumer_uid" : "VEHARBTVB2013032043", "depot_uid" : "NA", "station_uid" : "NA", "container_uid" : "NA", "worker_uid" : "0", "event_name" : "Consumer F110 KPL above Maximum", "event_type" : "Consumer KPL Above Maximum", "description" : "Consumer F110 KPL of 4.045000 is above the maximum of 2.6.", "processed" : "1" }, { "record" : "544", "data_event_uid" : "2723115d-e478-432d-9587-9b3aa36f17d3", "udatetime" : "2013-06-02 11:55:00", "customer_uid" : "ARBTVB20130320", "consumer_uid" : "VEHARBTVB2013032043", "depot_uid" : "NA", "station_uid" : "NA", "container_uid" : "NA", "worker_uid" : "0", "event_name" : "Consumer F110 KPL below Minimum", "event_type" : "Consumer KPL Below Minimum", "description" : "Consumer F110 KPL of 0.000000 is below the minimum of 2.3.", "processed" : "1" }], "groups" : [{ "record" : "9", "consumer_group_uid" : "GP1ARBTVB20130320", "customer_uid" : "ARBTVB20130320", "name" : "Truck", "description" : null, "other" : null, "min_kpl" : null, "max_kpl" : null }], "types" : [{ "record" : "11", "consumer_type_uid" : "TP1ARBTVB20130320", "customer_uid" : "ARBTVB20130320", "name" : "Truck", "description" : null, "other" : null, "min_kpl" : null, "max_kpl" : null }] }{ "_id" : ObjectId("52378add6058a7c73e239ffd"), "record" : "94", "consumer_uid" : "VEHARBTVB2013032044", "customer_uid" : "ARBTVB20130320", "consumer_group_uid" : "GP1ARBTVB20130320", "consumer_type_uid" : "TP1ARBTVB20130320", "tag" : "307E72B523AA", "name" : "F108", "secondary_name" : "WYG932GP", "description" : "Please enter", "current_odometer" : "646698", "prevoius_odometer" : null, "capacity" : "1000", "daily_limit" : "1000", "lk_cpk" : "0.00", "current_limit" : null, "current_limit_date" : null, "current_hours" : null, "prevoius_hours" : null, "capture_type" : "O", "km_limit" : "500", "setup" : "KMHR%REASON%VOLUME%WORKER", "min_kpl" : "2.3", "max_kpl" : "2.6", "transaction" : [{ "record" : "349", "consumer_transaction_uid" : "ad3bf082-a8d1-11e2-b3d0-001c14012e75", "udatetime" : "2013-04-19 11:00:46", "consumer_uid" : "VEHARBTVB2013032044", "worker_uid" : "OP1ARBTVB201303209", "volume" : "26.568", "odometer" : "646699.000", "hours" : "0", "source" : "Internal", "location" : "Unknown", "price" : "26.57", "remarks" : "Normal Filling", "start_time" : "2013-04-19 11:00:04", "end_time" : "2013-04-19 11:07:31", "cpk" : "0.00", "kpl" : "0.00", "bypassed" : "NO", "auth_uid" : "Unknown", "restype" : "Normal", "trip_distance" : "0" }, { "record" : "377", "consumer_transaction_uid" : "2b9e5d05-ab50-11e2-b3d0-001c14012e75", "udatetime" : "2013-04-22 03:17:31", "consumer_uid" : "VEHARBTVB2013032044", "worker_uid" : "OP1ARBTVB201303209", "volume" : "203.072", "odometer" : "646699.000", "hours" : "0", "source" : "Internal", "location" : "Unknown", "price" : "203.07", "remarks" : "Normal Filling", "start_time" : "2013-04-22 03:16:39", "end_time" : "2013-04-22 03:24:37", "cpk" : "0.00", "kpl" : "0.00", "bypassed" : "NO", "auth_uid" : "Unknown", "restype" : "Normal", "trip_distance" : "0" }, { "record" : "413", "consumer_transaction_uid" : "ef256588-ad63-11e2-b3d0-001c14012e75", "udatetime" : "2013-04-25 06:35:12", "consumer_uid" : "VEHARBTVB2013032044", "worker_uid" : "OP1ARBTVB201303209", "volume" : "604.066", "odometer" : "646700.000", "hours" : "0", "source" : "Internal", "location" : "Unknown", "price" : "604.07", "remarks" : "Normal Filling", "start_time" : "2013-04-25 06:33:52", "end_time" : "2013-04-25 06:48:06", "cpk" : "-0.01", "kpl" : "-0.01", "bypassed" : "NO", "auth_uid" : "Unknown", "restype" : "Normal", "trip_distance" : "-4" }], "events" : [{ "record" : "384", "data_event_uid" : "8fa44959-2e7f-4de8-a18d-5d89dd8d913a", "udatetime" : "2013-05-26 11:55:02", "customer_uid" : "ARBTVB20130320", "consumer_uid" : "VEHARBTVB2013032044", "depot_uid" : "NA", "station_uid" : "NA", "container_uid" : "NA", "worker_uid" : "0", "event_name" : "Consumer F108 KPL below Minimum", "event_type" : "Consumer KPL Below Minimum", "description" : "Consumer F108 KPL of 1.656667 is below the minimum of 2.3.", "processed" : "1" }, { "record" : "545", "data_event_uid" : "a332e227-18e1-4474-b89c-b3d3fa61f310", "udatetime" : "2013-06-02 11:55:00", "customer_uid" : "ARBTVB20130320", "consumer_uid" : "VEHARBTVB2013032044", "depot_uid" : "NA", "station_uid" : "NA", "container_uid" : "NA", "worker_uid" : "0", "event_name" : "Consumer F108 KPL below Minimum", "event_type" : "Consumer KPL Below Minimum", "description" : "Consumer F108 KPL of -215.810000 is below the minimum of 2.3.", "processed" : "1" }, { "record" : "1360", "data_event_uid" : "55777a42-57dd-498d-af7e-3c89cb4d9d15", "udatetime" : "2013-06-16 11:55:01", "customer_uid" : "ARBTVB20130320", "consumer_uid" : "VEHARBTVB2013032044", "depot_uid" : "NA", "station_uid" : "NA", "container_uid" : "NA", "worker_uid" : "0", "event_name" : "Consumer F108 KPL below Minimum", "event_type" : "Consumer KPL Below Minimum", "description" : "Consumer F108 KPL of 1.475000 is below the minimum of 2.3.", "processed" : "1" }], "groups" : [{ "record" : "9", "consumer_group_uid" : "GP1ARBTVB20130320", "customer_uid" : "ARBTVB20130320", "name" : "Truck", "description" : null, "other" : null, "min_kpl" : null, "max_kpl" : null }], "types" : [{ "record" : "11", "consumer_type_uid" : "TP1ARBTVB20130320", "customer_uid" : "ARBTVB20130320", "name" : "Truck", "description" : null, "other" : null, "min_kpl" : null, "max_kpl" : null }] }{ "_id" : ObjectId("52378add6058a7c73e23a002"), "record" : "99", "consumer_uid" : "VEHARBTVB2013032049", "customer_uid" : "ARBTVB20130320", "consumer_group_uid" : "GP1ARBTVB20130320", "consumer_type_uid" : "TP1ARBTVB20130320", "tag" : "307E72C2847A", "name" : "F101", "secondary_name" : "WYV597GP", "description" : "Please enter", "current_odometer" : "652178", "prevoius_odometer" : null, "capacity" : "1000", "daily_limit" : "1000", "lk_cpk" : "0.00", "current_limit" : null, "current_limit_date" : null, "current_hours" : null, "prevoius_hours" : null, "capture_type" : "O", "km_limit" : "500", "setup" : "KMHR%REASON%VOLUME%WORKER", "min_kpl" : "2.3", "max_kpl" : "2.6", "transaction" : [{ "record" : "293", "consumer_transaction_uid" : "4c63e3d9-8571-11e2-8602-000c29513585", "udatetime" : "2013-03-20 09:08:11", "consumer_uid" : "VEHARBTVB2013032049", "worker_uid" : "OP1ARBTVB201303224", "volume" : "4.000", "odometer" : "652179.000", "hours" : "0", "source" : "Internal", "location" : null, "price" : "4.00", "remarks" : "REASON 3", "start_time" : "2013-03-20 09:07:47", "end_time" : "2013-03-20 09:08:18", "cpk" : null, "kpl" : null, "bypassed" : null, "auth_uid" : null, "restype" : "diesel", "trip_distance" : "0" }, { "record" : "296", "consumer_transaction_uid" : "cee94825-8572-11e2-8602-000c29513585", "udatetime" : "2013-03-20 09:22:18", "consumer_uid" : "VEHARBTVB2013032049", "worker_uid" : "OP1ARBTVB201303230", "volume" : "11.000", "odometer" : "652180.000", "hours" : "0", "source" : "Internal", "location" : null, "price" : "11.00", "remarks" : "REASON 4", "start_time" : "2013-03-20 09:21:58", "end_time" : "2013-03-20 09:22:25", "cpk" : "0.09", "kpl" : "0.09", "bypassed" : null, "auth_uid" : null, "restype" : "diesel", "trip_distance" : "1" }, { "record" : "299", "consumer_transaction_uid" : "3be11d9a-8577-11e2-8602-000c29513585", "udatetime" : "2013-03-20 09:25:50", "consumer_uid" : "VEHARBTVB2013032049", "worker_uid" : "OP1ARBTVB201303201", "volume" : "49.000", "odometer" : "652181.000", "hours" : "0", "source" : "Internal", "location" : null, "price" : "49.00", "remarks" : "REASON 3", "start_time" : "2013-03-20 09:25:21", "end_time" : "2013-03-20 09:26:08", "cpk" : "0.02", "kpl" : "0.02", "bypassed" : null, "auth_uid" : null, "restype" : "diesel", "trip_distance" : "1" }], "events" : [{ "record" : "303", "data_event_uid" : "71a85cf0-eb54-4271-a9d4-e6f2a23bfc2a", "udatetime" : "2013-05-19 11:55:04", "customer_uid" : "ARBTVB20130320", "consumer_uid" : "VEHARBTVB2013032049", "depot_uid" : "NA", "station_uid" : "NA", "container_uid" : "NA", "worker_uid" : "0", "event_name" : "Consumer F101 KPL below Minimum", "event_type" : "Consumer KPL Below Minimum", "description" : "Consumer F101 KPL of 1.245000 is below the minimum of 2.3.", "processed" : "1" }, { "record" : "387", "data_event_uid" : "c38d054c-f6fc-4c06-a50e-18bbef3f8c77", "udatetime" : "2013-05-26 11:55:02", "customer_uid" : "ARBTVB20130320", "consumer_uid" : "VEHARBTVB2013032049", "depot_uid" : "NA", "station_uid" : "NA", "container_uid" : "NA", "worker_uid" : "0", "event_name" : "Consumer F101 KPL below Minimum", "event_type" : "Consumer KPL Below Minimum", "description" : "Consumer F101 KPL of 2.080000 is below the minimum of 2.3.", "processed" : "1" }, { "record" : "1922", "data_event_uid" : "e31f4b9e-9d1c-4f6e-bb04-220aede25351", "udatetime" : "2013-07-07 11:55:02", "customer_uid" : "ARBTVB20130320", "consumer_uid" : "VEHARBTVB2013032049", "depot_uid" : "NA", "station_uid" : "NA", "container_uid" : "NA", "worker_uid" : "0", "event_name" : "Consumer F101 KPL below Minimum", "event_type" : "Consumer KPL Below Minimum", "description" : "Consumer F101 KPL of 2.253333 is below the minimum of 2.3.", "processed" : "1" }], "groups" : [{ "record" : "9", "consumer_group_uid" : "GP1ARBTVB20130320", "customer_uid" : "ARBTVB20130320", "name" : "Truck", "description" : null, "other" : null, "min_kpl" : null, "max_kpl" : null }], "types" : [{ "record" : "11", "consumer_type_uid" : "TP1ARBTVB20130320", "customer_uid" : "ARBTVB20130320", "name" : "Truck", "description" : null, "other" : null, "min_kpl" : null, "max_kpl" : null }] }{ "_id" : ObjectId("52378b196058a7c73e23a013"), "deviceinfo" : { "ISO" : "ISO7224023", "analog1" : "0", "analog2" : "1", "com1" : "1", "com2" : "0", "country" : "USA", "date" : "2009/05/30 10:58:24 AM", "imei" : "357541000234567", "manufacturer" : "Not Known", "patent" : "PAT2683", "type" : "ERROR", "version" : "1" } }{ "_id" : ObjectId("52378b196058a7c73e23a02a"), "deviceinfo" : { "ISO" : "ISO1540023", "analog1" : "0", "analog2" : "0", "com1" : "1", "com2" : "1", "country" : "CHI", "date" : "2012/02/10 08:05:24 AM", "imei" : "351777040332536", "manufacturer" : "Not Known", "patent" : "PAT6077", "type" : "ERROR", "version" : "0" } }{ "_id" : ObjectId("52378b196058a7c73e23a048"), "status" : { "analog1" : "1", "analog2" : "0", "com1" : "0", "com2" : "1", "date" : "2014/10/16 02:34:24 PM", "imei" : "351777040332536", "status" : "ER3", "type" : "ERROR" } }{ "_id" : ObjectId("52378b196058a7c73e23a04c"), "status" : { "analog1" : "1", "analog2" : "1", "com1" : "1", "com2" : "0", "date" : "2007/05/17 03:15:24 AM", "imei" : "351777040332536", "status" : "ER3", "type" : "ERROR" } }{ "_id" : ObjectId("52378b196058a7c73e23a051"), "deviceinfo" : { "ISO" : "ISO7224023", "analog1" : "1", "analog2" : "1", "com1" : "0", "com2" : "1", "country" : "USA", "date" : "2008/10/04 05:41:24 AM", "imei" : "357541000234567", "manufacturer" : "Not Known", "patent" : "PAT2683", "type" : "ERROR", "version" : "2" } }{ "_id" : ObjectId("52378b196058a7c73e23a062"), "DA877546757 19B7047" : "0x000858924825760827744985894756780082774498590366174008277449858844138120827744985905275080082774498589343876008 27

```
74498589514965408277449859073274820827744985885832518082774498588359508208277449859061215660827744 9" }{ "_id" : ObjectId("52378b196058a7c73e23a068"), "packet" : [{ "" : "0x0008591650435608257448858880990900825744885875389746082574488590694051208257448858744881620825744885901560186082574488588639761608257448858899093020825744885907361778082574488589562609008257448858794617660825744 8" }] }{ "_id" : ObjectId("52378b196058a7c73e23a081"), "packet" : [{ "" : "0x0008590273925808147442859108206280814744285891439854081474428588195282808147442858855324220814744285887896584081474428591649793008147442858826360660814744285897866316081474428589561131208147442859148802040814744 2" }] }{ "_id" : ObjectId("52378b196058a7c73e23a094"), "transaction" : { "BIN" : "8588284676808047436", "date" : "2013/07/06 04:57:24 AM", "events" : [{ "date" : "2013/07/06 04:57:24 AM", "message" : "<message>", "type" : "NOTI" }, null], "imei" : "357023006702204", "readings" : [{ "date" : "2013/07/06 04:57:24 AM", "level" : "24363" }, null, { "date" : "2013/07/06 04:57:24 AM", "level" : "24363" }, { "date" : "2013/07/06 04:57:24 AM", "level" : "24363" }], "transactions" : [{ "date" : "2013/07/06 04:57:24 AM", "id" : "<id>", "odo" : "1000102", "uid" : "cafb30185f224dff8bb86be6ab804c08", "volume" : "8135" }, { "date" : "2013/07/06 04:57:24 AM", "id" : "<id>", "odo" : "1000102", "uid" : "cafb30185f224dff8bb86be6ab804c08", "volume" : "8135" }, { "date" : "2013/07/06 04:57:24 AM", "id" : "<id>", "odo" : "1000102", "uid" : "cafb30185f224dff8bb86be6ab804c08", "volume" : "8135" }], "version" : "0" } }{ "_id" : ObjectId("52378b196058a7c73e23a095"), "packet" : [{ "" : "0x0008587465850808037436859008753940803743685916701834080374368590506481008037436858933490780803743685880790628080374368589097394808037436859153356880803743685910678008080374368587510054008037436858769409080803743 6" }] }{ "_id" : ObjectId("52378b196058a7c73e23a097"), "status" : { "analog1" : "0", "analog2" : "1", "com1" : "0", "com2" : "0", "date" : "2002/11/22 05:32:24 AM", "imei" : "351777040332536", "status" : "ER3", "type" : "ERROR" } }{ "_id" : ObjectId("52378b196058a7c73e23a09b"), "" : "0x0008591126837208007434858829159660800743485876923724080074348590993386408007434859070861740800743485905319192080074348588166316080074348591188496808007434858996058900800743485895526166080074348590156763808007434 6" }{ "_id" : ObjectId("52378b196058a7c73e23a0a7"), "" : "0x0008591684716607957431858846940960795743185879414792079574318587911581207957431859120879780795743185888922746079574318591282221007957431859000546060795743185892756488079574318588542930607957431858922267420795743 1" }{ "_id" : ObjectId("52378b196058a7c73e23a0b1"), "AA87867272915956866" : "0x0008589444433607887427859117482520788742785876206712078874278591677420007887427858833299780788742785873462940788742785886342524078874278589459442007887427858927359680788742785909158544078874278591339632607887427 7" }{ "_id" : ObjectId("52378b196058a7c73e23a0b3"), "packet" : [{ "" : "0x0008587568822207887427859005956080788742785909573324078874278588228157407887427859050074440788742785898412526078874278597104658078874278588854494407887427859018312720788742785913304178078874278591518123007887427 7" }] }{ "_id" : ObjectId("52378b196058a7c73e23a0b4"), "" : "0x0008587744596407877427858851041180787742785897520074078774278589547588007877427858852604240787742785868095520787742785874391316078774278589514827407877427859095834100787742785891867192078774278589431635607877427 7" }{ "_id" : ObjectId("52378b196058a7c73e23a0bf"), "0x88753585B8837030" : "0x0008591331947207827424859093366300782742485908364168078274248587752157607827424859158731020782742485881856246078274248589953062607827424858931077100782742485881734460782742485898477848078274248590701367607827424 7" }{ "_id" : ObjectId("52378b196058a7c73e23a0d1"), "packet" : [{ "" : "0x0008591411024807737419858910029880773741985881123220077374198591532972407737419858914249020773741985878656608077374198589204614807737419858832981060773741985892925244077374198589043561607737419858987922000773741 9" }] }{ "_id" : ObjectId("52378b196058a7c73e23a0ec"), "packet" : [{ "" : "0x0008590070821607597411858747008140759741185907092660759741185882043218075974118590475015207597411859014725020759741185830724640759741185900477678075974118591654565407597411858874444480759741185898142028075974 1" }] }{ "_id" : ObjectId("52378b196058a7c73e23a0fb"), "transaction" : { "BHEX" : "0x0008591341273007527407858950799280752740785898865172075274 07", "date" : "2013/09/17 04:24:24 PM", "id" : "<id>", "imei" : "357023006714589", "odometer" : "<odometer>", "uid" : "59736298f7bb46d0b2d1fdb6a937fc9a", "version" : "0", "volume" : "8376" } }{ "_id" : ObjectId("52378b196058a7c73e23a0fe"), "CA889976795D666735" : "0x0008591078314607507405858937523440750740585913901094075074058588094958607507405858882514240750740585908607006075074058592720314075074058589763578407507405858853027180750740585893400138075074058589880420007507405 7" }{ "_id" : ObjectId("52378b196058a7c73e23a106"), "deviceinfo" : { "ISO" : "ISO7224023", "analog1" : "1", "analog2" : "1", "com1" : "0", "com2" : "0", "country" : "JAP", "date" : "2010/09/27 10:27:24 AM", "imei" : "357023006714589", "manufacturer" : "Not Known", "patent" : "PAT8015", "type" : "NOTI", "version" : "0" } }{ "_id" : ObjectId("52378b196058a7c73e23a10b"), "" : "0x0008590195118807457403858850931800745740385900325500074574038591198648207457403859072606060745740385872856680745740385855938020745740385908907414074574038587489375607457403858770091520745740385910911660074574037 3" }{ "_id" : ObjectId("52378b196058a7c73e23a11b"), "status" : { "analog1" : "1", "analog2" : "0", "com1" : "1", "com2" : "1", "date" : "2014/12/25 12:53:24 PM", "imei" : "351777040332536", "status" : "ER12", "type" : "ERROR" } }{ "_id" : ObjectId("52378b196058a7c73e23a11c"), "packet" : [{ "" : "0x0008589006429407357397858764994280735739785905445432073573978588451022007357397858937163920735739785912776208073573978589727662080735739785882627078073573978588949850607357397859140764500735739785916907232073573978589285781007357397 7" }] }{ "_id" : ObjectId("52378b196058a7c73e23a11e"), "transaction" : { "BIN" : "8588284676807347396", "date" : "2007/04/30 10:10:24 PM", "events" : [{ "date" : "2007/04/30 10:10:24 PM", "message" : "<message>", "type" : "ERROR" }, null], "imei" : "351777040332536", "readings" : [{ "date" : "2007/04/30 10:10:24 PM", "level" : "47529" }, null, { "date" : "2007/04/30 10:10:24 PM", "level" : "47529" }, { "date" : "2007/04/30 10:10:24 PM", "level" : "47529" }], "transactions" : [{ "date" : "2007/04/30 10:10:24 PM", "id" : "<id>", "odo" : "1002209", "uid" : "029c810441ee463ba67a1dea80c472cd", "volume" : "4845" }, { "date" : "2007/04/30 10:10:24 PM", "id" : "<id>", "odo" : "1002209", "uid" : "029c810441ee463ba67a1dea80c472cd", "volume" : "4845" }, { "date" : "2007/04/30 10:10:24 PM", "id" : "<id>", "odo" : "1002209", "uid" : "029c810441ee463ba67a1dea80c472cd", "volume" : "4845" }], "version" : "3" } }{ "_id" : ObjectId("52378b196058a7c73e23a122"), "packet" : [{ "" : "0x0008591404719407327395858760734640732739585916343814073273958588086169807327395858843590680732739585916655255239073273958588568848207327395858991865620732739585916082640073273958591253979607327395 7" }] }{ "_id" : ObjectId("52378b196058a7c73e23a124"), "status" : { "analog1" : "1", "analog2" : "0", "com1" : "0", "com2" : "0", "date" : "2003/10/16 04:12:24 PM", "imei" : "351777040332536", "status" : "ER12", "type" : "ERROR" } }{ "_id" : ObjectId("52378b196058a7c73e23a125"), "" : "0x0008589233948807317395858880711780731739585916973700073173958588785597607317395859097642500731739585916430268073173958589335068007317395859126523860731739585901168026073173958589218209007317395859002938560731739 5" }{ "_id" : ObjectId("52378b196058a7c73e23a129"), "" : "0x0008591222401007297393858882818680729739385899424204072973938588949061607297393858893645440729739385897526944072973938583376580072973938590878440807297393859115951380729739385913540176072973938588708209007297393 3" }{ "_id" : ObjectId("52378b196058a7c73e23a12b"), "packet" : [{ "" : "0x0008590086752807277392858868055800727739285908893176072773928591362437407277392858987538120727739285908787702072773928589215279207277392858868122280727739285910562826072773928591691307007277392858931653040727739 2" }] }{ "_id" : ObjectId("52378b196058a7c73e23a12f"), "transaction" : { "BIN" : "8588284676807257391", "date" : "2015/03/12 03:56:24 AM", "events" : [{ "date" : "2015/03/12 03:56:24 AM", "message" : "<message>", "type" : "ERROR" }, null], "imei" : "357541000234567", "readings" : [{ "date" : "2015/03/12 03:56:24 AM", "level" : "42541" }, null, { "date" : "2015/03/12 03:56:24 AM", "level" : "42541" }, { "date" : "2015/03/12 03:56:24 AM", "level" : "42541" }], "transactions" : [{ "date" : "2015/03/12 03:56:24 AM", "id" : "<id>", "odo" : "1001764", "uid" : "c18f472a25c844bd92d7c614e9c26a22", "volume" : "1073" }, { "date" : "2015/03/12 03:56:24 AM", "id" : "<id>", "odo" : "1001764", "uid" : "c18f472a25c844bd92d7c614e9c26a22", "volume" : "1073" }, { "date" : "2015/03/12 03:56:24 AM", "id" : "<id>", "odo" : "1001764", "uid" : "c18f472a25c844bd92d7c614e9c26a22", "volume" : "1073" }], "version" : "4" } }{ "_id" : ObjectId("52378b196058a7c73e23a13b"), "transaction" : { "BIN" : "8588284676807177387", "date" : "2004/02/01 02:39:24 AM", "events" : [{ "date" : "2004/02/01 02:39:24 AM", "message" : "<message>", "type" : "ERROR" }, null], "imei" : "357541000234567", "readings" : [{ "date" : "2004/02/01 02:39:24 AM", "level" : "27466" }, null, { "date" : "2004/02/01 02:39:24 AM", "level" : "27466" }, { "date" : "2004/02/01 02:39:24 AM", "level" : "27466" }], "transactions" : [{ "date" : "2004/02/01 02:39:24 AM", "id" : "<id>", "odo" : "1001337", "uid" : "a2e699becdeb432ea32606c0108e404a", "volume" : "3731" }, { "date" : "2004/02/01
```

02:39:24 AM", "id" : "<id>", "odo" : "1001337", "uid" : "a2e699becdeb432ea32606c0108e404a", "volume" : "3731" }, { "date" : "2004/02/01 02:39:24 AM", "id" : "<id>", "odo" : "1001337", "uid" : "a2e699becdeb432ea32606c0108e404a", "volume" : "3731" }], "version" : "2" } }{ "_id" : ObjectId("52378b196058a7c73e23a13e"), "status" : { "analog1" : "0", "analog2" : "1", "com1" : "1", "com2" : "1", "date" : "2010/12/04 03:02:24 AM", "imei" : "357541000234567", "status" : "Offline", "type" : "ERROR" } }{ "_id" : ObjectId("52378b196058a7c73e23a13f"), "" : "0x0008590947379007157385859166286700715738585887529318071573858589738680207157385859054720780715738585880181448071573858590387059407157385859058206000715738585888579756071573858586655425207157385859009767460715738 5" }{ "_id" : ObjectId("52378b196058a7c73e23a14c"), "" : "0x0008588505920207087381858796490200708738185884785224070873818588595703007087381858974758000708738185886898106070873818588078024807087381859140808060708738185880661364070873818588363131607087381859059671140708738 1" }{ "_id" : ObjectId("52378b196058a7c73e23a16c"), "packet" : [{ "" : "0x0008588996573806897371859091469580689737185874704498068973718588316554806897371858789866920689737185894168864068973718591019492406897371859056352420689737185913946568068973718590560894406897371858997671220689737 1" }] }{ "_id" : ObjectId("52378b196058a7c73e23a178"), "packet" : [{ "" : "0x0008587749519406807365858993977860680736585889174002068073658588851987606807365858907420122068073658587541102406807365859165040740680736585900795144068073658589347490406807365858819895060680736585907380366068073 5" }] }{ "_id" : ObjectId("52378b196058a7c73e23a190"), "" : "0x0008591717510806667357859125208180666735785910045896066673578588875208206667357859056942760666735785882594888066673575858831451660666735785894343080066673578591478614806667357858761824600666735785888964380066673 7" }{ "_id" : ObjectId("52378b196058a7c73e23a192"), "transaction" : { "BIN" : "8588284676806647356", "date" : "2014/06/23 02:30:24 PM", "events" : [{ "date" : "2014/06/23 02:30:24 PM", "message" : "<message>", "type" : "NOTI" }, null], "imei" : "357023006702204", "readings" : [{ "date" : "2014/06/23 02:30:24 PM", "level" : "23768" }, null, { "date" : "2014/06/23 02:30:24 PM", "level" : "23768" }, { "date" : "2014/06/23 02:30:24 PM", "level" : "23768" }], "transactions" : [{ "date" : "2014/06/23 02:30:24 PM", "id" : "<id>", "odo" : "1003043", "uid" : "cf3332c35afe49e29626cf86410c8fbb", "volume" : "4334" }, { "date" : "2014/06/23 02:30:24 PM", "id" : "<id>", "odo" : "1003043", "uid" : "cf3332c35afe49e29626cf86410c8fbb", "volume" : "4334" }, { "date" : "2014/06/23 02:30:24 PM", "id" : "<id>", "odo" : "1003043", "uid" : "cf3332c35afe49e29626cf86410c8fbb", "volume" : "4334" }], "version" : "2" } }{ "_id" : ObjectId("52378b196058a7c73e23a197"), "transaction" : { "BIN" : "8588284676806607354", "date" : "2011/01/13 08:02:24 AM", "events" : [{ "date" : "2011/01/13 08:02:24 AM", "message" : "<message>", "type" : "ERROR" }, null], "imei" : "357541000234567", "readings" : [{ "date" : "2011/01/13 08:02:24 AM", "level" : "31156" }, null, { "date" : "2011/01/13 08:02:24 AM", "level" : "31156" }, { "date" : "2011/01/13 08:02:24 AM", "level" : "31156" }], "transactions" : [{ "date" : "2011/01/13 08:02:24 AM", "id" : "<id>", "odo" : "1003137", "uid" : "c51bb75fde7b4203804d42cfd97f9978", "volume" : "8498" }, { "date" : "2011/01/13 08:02:24 AM", "id" : "<id>", "odo" : "1003137", "uid" : "c51bb75fde7b4203804d42cfd97f9978", "volume" : "8498" }, { "date" : "2011/01/13 08:02:24 AM", "id" : "<id>", "odo" : "1003137", "uid" : "c51bb75fde7b4203804d42cfd97f9978", "volume" : "8498" }], "version" : "3" } }{ "_id" : ObjectId("52378b196058a7c73e23a19c"), "status" : { "analog1" : "1", "analog2" : "0", "com1" : "1", "com2" : "0", "date" : "2011/02/18 05:58:24 AM", "imei" : "357541000234567", "status" : "Online", "type" : "ERROR" } }{ "_id" : ObjectId("52378b196058a7c73e23a19d"), "transaction" : { "BIN" : "8588284676806567352", "date" : "2016/01/24 06:59:24 AM", "events" : [{ "date" : "2016/01/24 06:59:24 AM", "message" : "<message>", "type" : "ERROR" }, null], "imei" : "357541000234567", "readings" : [{ "date" : "2016/01/24 06:59:24 AM", "level" : "36288" }, null, { "date" : "2016/01/24 06:59:24 AM", "level" : "36288" }, { "date" : "2016/01/24 06:59:24 AM", "level" : "36288" }], "transactions" : [{ "date" : "2016/01/24 06:59:24 AM", "id" : "<id>", "odo" : "1000909", "uid" : "424c571ea0f649ff90777e6560044a22", "volume" : "6979" }, { "date" : "2016/01/24 06:59:24 AM", "id" : "<id>", "odo" : "1000909", "uid" : "424c571ea0f649ff90777e6560044a22", "volume" : "6979" }, { "date" : "2016/01/24 06:59:24 AM", "id" : "<id>", "odo" : "1000909", "uid" : "424c571ea0f649ff90777e6560044a22", "volume" : "6979" }], "version" : "3" } }{ "_id" : ObjectId("52378b196058a7c73e23a1a9"), "DA877314887F946922" : "0x0008588444711206487347859006945360648734785914415540064873478591316464206487347858810636580648734785914384382064873478591604949006487347858805493200648734785894129126064873478587749559606487347858801726580648734 7" }{ "_id" : ObjectId("52378b196058a7c73e23a1b0"), "transaction" : { "BHEX" : "0x0008589910986406447345859030241320644734585877704582064473 45", "date" : "2012/07/14 04:14:24 AM", "id" : "<id>", "imei" : "357023006714589", "odometer" : "<odometer>", "uid" : "3cb996a0cc034f78a041563686655955", "version" : "2", "volume" : "1342" } }{ "_id" : ObjectId("52378b196058a7c73e23a1b7"), "CA889976795D666735" : "0x0008590332637006387341859168010740638734185902548524063873418589301325806387341859064617180638734185885552318063873418589007511160638734185892896360063873418589301183606387341858871903406387341859116977020638734 1" }{ "_id" : ObjectId("52378b196058a7c73e23a1b8"), "deviceinfo" : { "ISO" : "ISO7224023", "analog1" : "0", "analog2" : "1", "com1" : "0", "com2" : "1", "country" : "JAP", "date" : "2008/11/11 05:16:24 PM", "imei" : "357023006714589", "manufacturer" : "Not Known", "patent" : "PAT8015", "type" : "NOTI", "version" : "1" } }{ "_id" : ObjectId("52378b196058a7c73e23a1ba"), "packet" : [{ "" : "0x0008588220040606367340859157110360636734085901924068063673408589697120006367340858759410606367340858789441700636734085911872320063673408588703069406367340858977332960636734085890461614063673408589260691406367340 0" }] }{ "_id" : ObjectId("52378b196058a7c73e23a1be"), "deviceinfo" : { "ISO" : "ISO7224023", "analog1" : "0", "analog2" : "0", "com1" : "0", "com2" : "1", "country" : "JAP", "date" : "2012/12/24 12:55:24 PM", "imei" : "357023006714589", "manufacturer" : "Not Known", "patent" : "PAT8015", "type" : "NOTI", "version" : "4" } }{ "_id" : ObjectId("52378b196058a7c73e23a1d3"), "packet" : [{ "" : "0x0008591476484806187330858751286620618733085915459325406187330858917687440618733085911986272061873308587529014606187330858977040580618733085902628654061873308588784643006187330859050590140618733085916625736061873 0" }] }{ "_id" : ObjectId("52378b196058a7c73e23a1d5"), "transaction" : { "BHEX" : "0x0008587538801806177329859146395080617732985903 8387606177329", "date" : "2015/10/21 12:19:24 AM", "id" : "<id>", "imei" : "351777040332536", "odometer" : "<odometer>", "uid" : "5c443e7ccd564dff849964049532a998", "version" : "3", "volume" : "2181" } }{ "_id" : ObjectId("52378b196058a7c73e23a1df"), "packet" : [{ "" : "0x0008587546929406107325859024269580610732585913187556061073258588898337006107325859105825600610732585901190400610732585888182688061073258588724618406107325858908276920610732585911588562061073258590701284806107325 5" }] }{ "_id" : ObjectId("52378b196058a7c73e23a1f2"), "transaction" : { "BIN" : "8588284676805967317", "date" : "2010/08/08 09:12:24 AM", "events" : [{ "date" : "2010/08/08 09:12:24 AM", "message" : "<message>", "type" : "ERROR" }, null], "imei" : "351777040332536", "readings" : [{ "date" : "2010/08/08 09:12:24 AM", "level" : "28860" }, null, { "date" : "2010/08/08 09:12:24 AM", "level" : "28860" }, { "date" : "2010/08/08 09:12:24 AM", "level" : "28860" }], "transactions" : [{ "date" : "2010/08/08 09:12:24 AM", "id" : "<id>", "odo" : "1000775", "uid" : "eef059b45c4e4338bb9cf9521442361d", "volume" : "6421" }, { "date" : "2010/08/08 09:12:24 AM", "id" : "<id>", "odo" : "1000775", "uid" : "eef059b45c4e4338bb9cf9521442361d", "volume" : "6421" }, { "date" : "2010/08/08 09:12:24 AM", "id" : "<id>", "odo" : "1000775", "uid" : "eef059b45c4e4338bb9cf9521442361d", "volume" : "6421" }], "version" : "4" } }{ "_id" : ObjectId("52378b196058a7c73e23a1f6"), "deviceinfo" : { "ISO" : "ISO1540023", "analog1" : "0", "analog2" : "1", "com1" : "1", "com2" : "0", "country" : "CHI", "date" : "2013/09/03 05:52:24 PM", "imei" : "351777040332536", "manufacturer" : "Not Known", "patent" : "PAT6077", "type" : "ERROR", "version" : "3" } }{ "_id" : ObjectId("52378b196058a7c73e23a1f7"), "deviceinfo" : { "ISO" : "ISO1540023", "analog1" : "1", "analog2" : "0", "com1" : "1", "com2" : "0", "country" : "CHI", "date" : "2012/12/08 03:30:24 AM", "imei" : "351777040332536", "manufacturer" : "Not Known", "patent" : "PAT6077", "type" : "ERROR", "version" : "3" } }{ "_id" : ObjectId("52378b196058a7c73e23a1fb"), "status" : { "analog1" : "1", "analog2" : "1", "com1" : "0", "com2" : "0", "date" : "2012/09/06 11:07:24 AM", "imei" : "351777040332536", "status" : "ER34", "type" : "ERROR" } }{ "_id" : ObjectId("52378b196058a7c73e23a212"), "" : "0x0008590954928205707302859060916200570730285915825084057073028587937974605707302858790267600570730285879203160057073028590660897005707302858941185360570730285878605284057073028591232630405707302858879552520570730 2" }{ "_id" : ObjectId("52378b196058a7c73e23a217"), "deviceinfo" : { "ISO" : "ISO3910010", "analog1" : "1", "analog2" : "0", "com1" : "1", "com2" : "0", "country" : "USA", "date" : "2008/03/14 04:12:24 PM", "imei" : "357023006702204", "manufacturer" : "Not Known", "patent" : "PAT4278", "type" : "NOTI", "version" : "0" } }{ "_id" : ObjectId("52378b196058a7c73e23a21b"), "status" : { "analog1" : "0", "analog2" : "1", "com1" : "1", "com2" : "0", "date" : "2013/09/06 04:13:24 PM", "imei" : "357023006702204", "status" : "ER68", "type" : "NOTI" } }{ "_id" : ObjectId("52378b196058a7c73e23a233"), "DA877314887F946922" : "0x0008589571826805447288858834711700544728885906193980544728885881087778054472888591691074805447288858923206000544

72888589439803405447288858954061240544728885900470526054472888589731868405447288858950962720544728 8" }{ "_id" : ObjectId("52378b196058a7c73e23a247"), "transaction" : { "BIN" : "8588284676805267277", "date" : "2011/02/19 12:47:24 AM", "events" : [{ "date" : "2011/02/19 12:47:24 AM", "message" : "<message>", "type" : "ERROR" }, null], "imei" : "351777040332536", "readings" : [{ "date" : "2011/02/19 12:47:24 AM", "level" : "3632" }, null, { "date" : "2011/02/19 12:47:24 AM", "level" : "3632" }, { "date" : "2011/02/19 12:47:24 AM", "level" : "3632" }], "transactions" : [{ "date" : "2011/02/19 12:47:24 AM", "id" : "<id>", "odo" : "1004213", "uid" : "6b5b27ef3f3a4340920f1f451d188449", "volume" : "1146" }, { "date" : "2011/02/19 12:47:24 AM", "id" : "<id>", "odo" : "1004213", "uid" : "6b5b27ef3f3a4340920f1f451d188449", "volume" : "1146" }, { "date" : "2011/02/19 12:47:24 AM", "id" : "<id>", "odo" : "1004213", "uid" : "6b5b27ef3f3a4340920f1f451d188449", "volume" : "1146" }], "version" : "1" } }{ "_id" : ObjectId("52378b196058a7c73e23a252"), "transaction" : { "BIN" : "8588284676805167272", "date" : "2011/05/12 11:54:24 AM", "events" : [{ "date" : "2011/05/12 11:54:24 AM", "message" : "<message>", "type" : "ERROR" }, null], "imei" : "357541000234567", "readings" : [{ "date" : "2011/05/12 11:54:24 AM", "level" : "25851" }, null, { "date" : "2011/05/12 11:54:24 AM", "level" : "25851" }, { "date" : "2011/05/12 11:54:24 AM", "level" : "25851" }], "transactions" : [{ "date" : "2011/05/12 11:54:24 AM", "id" : "<id>", "odo" : "1001969", "uid" : "1eb4c10fe22a41f9ac162b39fe635052", "volume" : "8970" }, { "date" : "2011/05/12 11:54:24 AM", "id" : "<id>", "odo" : "1001969", "uid" : "1eb4c10fe22a41f9ac162b39fe635052", "volume" : "8970" }, { "date" : "2011/05/12 11:54:24 AM", "id" : "<id>", "odo" : "1001969", "uid" : "1eb4c10fe22a41f9ac162b39fe635052", "volume" : "8970" }], "version" : "4" } }{ "_id" : ObjectId("52378b196058a7c73e23a256"), "transaction" : { "BIN" : "8588284676805137270", "date" : "2007/08/05 12:50:24 PM", "events" : [{ "date" : "2007/08/05 12:50:24 PM", "message" : "<message>", "type" : "ERROR" }, null], "imei" : "357541000234567", "readings" : [{ "date" : "2007/08/05 12:50:24 PM", "level" : "31764" }, null, { "date" : "2007/08/05 12:50:24 PM", "level" : "31764" }, { "date" : "2007/08/05 12:50:24 PM", "level" : "31764" }], "transactions" : [{ "date" : "2007/08/05 12:50:24 PM", "id" : "<id>", "odo" : "1004008", "uid" : "743b5e3274344b50ad19562faa9482f2", "volume" : "3144" }, { "date" : "2007/08/05 12:50:24 PM", "id" : "<id>", "odo" : "1004008", "uid" : "743b5e3274344b50ad19562faa9482f2", "volume" : "3144" }, { "date" : "2007/08/05 12:50:24 PM", "id" : "<id>", "odo" : "1004008", "uid" : "743b5e3274344b50ad19562faa9482f2", "volume" : "3144" }], "version" : "0" } }{ "_id" : ObjectId("52378b196058a7c73e23a258"), "packet" : [{ "" : "0x0008591555321805117269858994242640511726985889439358051172698588263928805117269858976777605117269858773225980511726985907710450051172698590829811405117269858890148040511726985887980200051172698589752028405117269" }] }{ "_id" : ObjectId("52378b196058a7c73e23a25f"), "deviceinfo" : { "ISO" : "ISO3910010", "analog1" : "1", "analog2" : "0", "com1" : "1", "com2" : "1", "country" : "USA", "date" : "2014/08/27 02:47:24 AM", "imei" : "357023006702204", "manufacturer" : "Not Known", "patent" : "PAT4278", "type" : "NOTI", "version" : "1" } }{ "_id" : ObjectId("52378b196058a7c73e23a260"), "packet" : [{ "" : "0x0008588183984205047265859140786280504726585879732930050472658587907041005047265858755653720504726585895166904050472658590014627405047265859130381260504726585898183548050472658591726163405047265858943744240504726 5" }] }{ "_id" : ObjectId("52378b196058a7c73e23a268"), "transaction" : { "BIN" : "8588284676804967260", "date" : "2014/05/05 08:29:24 AM", "events" : [{ "date" : "2014/05/05 08:29:24 AM", "message" : "<message>", "type" : "NOTI" }, null], "imei" : "357023006702204", "readings" : [{ "date" : "2014/05/05 08:29:24 AM", "level" : "6053" }, null, { "date" : "2014/05/05 08:29:24 AM", "level" : "6053" }, { "date" : "2014/05/05 08:29:24 AM", "level" : "6053" }], "transactions" : [{ "date" : "2014/05/05 08:29:24 AM", "id" : "<id>", "odo" : "1002909", "uid" : "c64bbba74d654ffa8e04d4982dab6300", "volume" : "7379" }, { "date" : "2014/05/05 08:29:24 AM", "id" : "<id>", "odo" : "1002909", "uid" : "c64bbba74d654ffa8e04d4982dab6300", "volume" : "7379" }, { "date" : "2014/05/05 08:29:24 AM", "id" : "<id>", "odo" : "1002909", "uid" : "c64bbba74d654ffa8e04d4982dab6300", "volume" : "7379" }], "version" : "3" } }{ "_id" : ObjectId("52378b196058a7c73e23a281"), "packet" : [{ "" : "0x0008591077123604757248858804515800475724885887968032047572488590689722047572488591331312404757248859059673004757248859027565380475724885882475062047572488597107934047572488591531242004757248859053477640475724 8" }] }{ "_id" : ObjectId("52378b196058a7c73e23a292"), "packet" : [{ "" : "0x0008590461297404607240858972751780460724085897264162046072408589433655204607240858886529740460724085877662876046072408591409919604607240858956682100460724085912588582046072408587460193404607240859081897900460724 0" }] }{ "_id" : ObjectId("52378b196058a7c73e23a29b"), "transaction" : { "BIN" : "8588284676804517234", "date" : "2011/08/28 01:28:25 PM", "events" : [{ "date" : "2011/08/28 01:28:25 PM", "message" : "<message>", "type" : "ERROR" }, null], "imei" : "357541000234567", "readings" : [{ "date" : "2011/08/28 01:28:25 PM", "level" : "10000" }, null, { "date" : "2011/08/28 01:28:25 PM", "level" : "10000" }, { "date" : "2011/08/28 01:28:25 PM", "level" : "10000" }], "transactions" : [{ "date" : "2011/08/28 01:28:25 PM", "id" : "<id>", "odo" : "1000402", "uid" : "26a01ee6b603455d9f47d184fcffbc27", "volume" : "5207" }, { "date" : "2011/08/28 01:28:25 PM", "id" : "<id>", "odo" : "1000402", "uid" : "26a01ee6b603455d9f47d184fcffbc27", "volume" : "5207" }, { "date" : "2011/08/28 01:28:25 PM", "id" : "<id>", "odo" : "1000402", "uid" : "26a01ee6b603455d9f47d184fcffbc27", "volume" : "5207" }], "version" : "1" } }{ "_id" : ObjectId("52378b196058a7c73e23a29c"), "" : "0x0008587771888004517234858949971520451723485894971385752045172348589847903004517234858928024840451723485904851276045172348589047400451723485894351708045172348590461657404517234859039311520451723485890913318045172 4" }{ "_id" : ObjectId("52378b196058a7c73e23a2bb"), "packet" : [{ "" : "0x0008591540146004237218859107762520423721885890377182042372188591208387404237218858829245880423721885896567772042372188590269983204237218858992193040423721885898619214042372188590000117004237218858995876740423721 8" }] }{ "_id" : ObjectId("52378b196058a7c73e23a2c4"), "status" : { "analog1" : "0", "analog2" : "1", "com1" : "0", "com2" : "1", "date" : "2015/07/17 02:31:25 PM", "imei" : "351777040332536", "status" : "ER12", "type" : "ERROR" } }{ "_id" : ObjectId("52378b196058a7c73e23a2d1"), "" : "0x0008590510850204037207858959543200403720785890495424040372078591525694404037207858808563400403720785858905060403720785882052836040372078588226515204037207858961301320403720785889652892040372078590800077804037207" }{ "_id" : ObjectId("52378b196058a7c73e23a2d3"), "transaction" : { "BHEX" : "0x0008591108617004017206858804397300401720685891675660040172 06", "date" : "2016/03/11 08:24:25 AM", "id" : "<id>", "imei" : "357541000234567", "odomenter" : "<odomenter>", "uid" : "aa0d552820cd4866a96920e054e3972a", "version" : "0", "volume" : "1793" } }{ "_id" : ObjectId("52378b196058a7c73e23a2d9"), "packet" : [{ "" : "0x0008589755164003957202858883591060395720285878305776039572028591228827003957202859004030020395720285916505124039572028588468900039572028590780514203957202858763228480395720285881775162039572028589502241803957202 " }] }{ "_id" : ObjectId("52378b196058a7c73e23a2da"), "packet" : [{ "" : "0x0008589394804603957202859015153360395720285896655990039572028589640021603957202859000281640395720285878867670039572028590813190803957202859067945920395720285886240854039572028589502782403957202858765527603957202 " }] }{ "_id" : ObjectId("52378b196058a7c73e23a2df"), "deviceinfo" : { "ISO" : "ISO3910010", "analog1" : "1", "analog2" : "1", "com1" : "0", "com2" : "1", "country" : "USA", "date" : "2012/01/11 11:47:25 AM", "imei" : "357023006702204", "manufacturer" : "Not Known", "patent" : "PAT4278", "type" : "NOTI", "version" : "2" } }{ "_id" : ObjectId("52378b196058a7c73e23a2ea"), "status" : { "analog1" : "1", "analog2" : "1", "com1" : "0", "com2" : "0", "date" : "2011/07/04 01:42:25 AM", "imei" : "357023006714589", "status" : "ER105", "type" : "NOTI" } }{ "_id" : ObjectId("52378b196058a7c73e23a2ed"), "packet" : [{ "" : "0x0008588954522803777192858888158860377719285908899830037771928590282037203777192859001389960377719285901122270037771928589735637003777192859103842720377719285898794540037771928590179940003777192859112180920377719 2" }] }{ "_id" : ObjectId("52378b196058a7c73e23a2f4"), "status" : { "analog1" : "1", "analog2" : "0", "com1" : "1", "com2" : "0", "date" : "2011/05/23 03:30:25 AM", "imei" : "357023006714589", "status" : "ER105", "type" : "NOTI" } }{ "_id" : ObjectId("52378b196058a7c73e23a2f5"), "status" : { "analog1" : "0", "analog2" : "1", "com1" : "1", "com2" : "1", "date" : "2013/02/09 09:11:25 PM", "imei" : "357023006714589", "status" : "ER105", "type" : "NOTI" } }{ "_id" : ObjectId("52378b196058a7c73e23a301"), "deviceinfo" : { "ISO" : "ISO3910010", "analog1" : "0", "analog2" : "1", "com1" : "1", "com2" : "0", "country" : "USA", "date" : "2005/09/10 11:14:25 AM", "imei" : "357023006702204", "manufacturer" : "Not Known", "patent" : "PAT4278", "type" : "NOTI", "version" : "4" } }{ "_id" : ObjectId("52378b196058a7c73e23a313"), "deviceinfo" : { "ISO" : "ISO7224023", "analog1" : "1", "analog2" : "0", "com1" : "0", "com2" : "0", "country" : "USA", "date" : "2012/12/12 03:32:25 AM", "imei" : "357541000234567", "manufacturer" : "Not Known", "patent" : "PAT2683", "type" : "ERROR", "version" : "0" } }{ "_id" : ObjectId("52378b196058a7c73e23a320"), "CA889976795D666735" : "0x0008590742029003257162858758677060325716285910134960032571628588998467403257162858791487640325716285899330214032571628587607667403257162858829780060325716285874422954032571628589791133403257162858958864660325716 2" }{ "_id" : ObjectId("52378b196058a7c73e23a321"), "status" : { "analog1" : "0", "analog2" : "1", "com1" : "1", "com2" : "1", "date" : "2005/06/08 02:22:25 PM", "imei" : "357023006714589", "status" : "ER105", "type" : "NOTI" } }{ "_id" : ObjectId("52378b196058a7c73e23a322"), "packet" : [{ "" : "0x0008589534878803227161858894343000322716185909868428032271618588787106603227161859112512540322716185869648380322716185890262570032271618590954419403227161859024763920322716185900732498032271618587674776803227161 7" }] }{ "_id" :

ObjectId("52378b196058a7c73e23a323"), "" : "0x0008589473604403217160858918338680321716085881822592032171608587819445203217160858898361740321716085891344262032171608588210782603217160858920283580321716085897689190032171608590761494203217160858850014280321716 0" }{ "_id" : ObjectId("52378b196058a7c73e23a337"), "deviceinfo" : { "ISO" : "ISO3910010", "analog1" : "0", "analog2" : "1", "com1" : "0", "com2" : "1", "country" : "USA", "date" : "2011/08/18 07:37:25 AM", "imei" : "357023006702204", "manufacturer" : "Not Known", "patent" : "PAT4278", "type" : "NOTI", "version" : "3" } }{ "_id" : ObjectId("52378b196058a7c73e23a33c"), "status" : { "analog1" : "0", "analog2" : "1", "com1" : "0", "com2" : "1", "date" : "2007/09/21 01:41:25 AM", "imei" : "357023006702204", "status" : "ER1", "type" : "NOTI" } }{ "_id" : ObjectId("52378b196058a7c73e23a343"), "deviceinfo" : { "ISO" : "ISO1540023", "analog1" : "0", "analog2" : "1", "com1" : "1", "com2" : "0", "country" : "CHI", "date" : "2004/09/17 02:20:25 PM", "imei" : "351777040332536", "manufacturer" : "Not Known", "patent" : "PAT6077", "type" : "ERROR", "version" : "0" } }{ "_id" : ObjectId("52378b196058a7c73e23a345"), "transaction" : { "BIN" : "8588284676802847139", "date" : "2015/12/14 08:55:25 AM", "events" : [{ "date" : "2015/12/14 08:55:25 AM", "message" : "<message>", "type" : "ERROR" }, null], "imei" : "351777040332536", "readings" : [{ "date" : "2015/12/14 08:55:25 AM", "level" : "41294" }, null, { "date" : "2015/12/14 08:55:25 AM", "level" : "41294" }, { "date" : "2015/12/14 08:55:25 AM", "level" : "41294" }], "transactions" : [{ "date" : "2015/12/14 08:55:25 AM", "id" : "<id>", "odo" : "1002688", "uid" : "8b45ca7401494cc985ef334086850b93", "volume" : "666" }, { "date" : "2015/12/14 08:55:25 AM", "id" : "<id>", "odo" : "1002688", "uid" : "8b45ca7401494cc985ef334086850b93", "volume" : "666" }, { "date" : "2015/12/14 08:55:25 AM", "id" : "<id>", "odo" : "1002688", "uid" : "8b45ca7401494cc985ef334086850b93", "volume" : "666" }], "version" : "3" } }{ "_id" : ObjectId("52378b196058a7c73e23a34c"), "packet" : [{ "" : "0x0008589437704002777135858870978160277713585898492704027771358590210127802777135858755294200277713585901289838027771358589013822602777135859053443380277713585907568940027771358588089562802777135859137171580277713 5" }] }{ "_id" : ObjectId("52378b196058a7c73e23a350"), "transaction" : { "BHEX" : "0x0008591520721602737133858938325040273713385896284176027371 3", "date" : "2003/08/25 12:44:25 PM", "id" : "<id>", "imei" : "351777040332536", "odometer" : "<odometer>", "uid" : "e07e8115fd894e56a07747c303d12677", "version" : "1", "volume" : "3977" } }{ "_id" : ObjectId("52378b196058a7c73e23a35d"), "transaction" : { "BIN" : "8588284676802577123", "date" : "2013/03/07 10:09:25 AM", "events" : [{ "date" : "2013/03/07 10:09:25 AM", "message" : "<message>", "type" : "NOTI" }, null], "imei" : "357023006714589", "readings" : [{ "date" : "2013/03/07 10:09:25 AM", "level" : "17685" }, null, { "date" : "2013/03/07 10:09:25 AM", "level" : "17685" }, { "date" : "2013/03/07 10:09:25 AM", "level" : "17685" }], "transactions" : [{ "date" : "2013/03/07 10:09:25 AM", "id" : "<id>", "odo" : "1003010", "uid" : "87f9287f1e2548a386d415d92fc49e5e", "volume" : "5153" }, { "date" : "2013/03/07 10:09:25 AM", "id" : "<id>", "odo" : "1003010", "uid" : "87f9287f1e2548a386d415d92fc49e5e", "volume" : "5153" }, { "date" : "2013/03/07 10:09:25 AM", "id" : "<id>", "odo" : "1003010", "uid" : "87f9287f1e2548a386d415d92fc49e5e", "volume" : "5153" }], "version" : "1" } }{ "_id" : ObjectId("52378b196058a7c73e23a36d"), "" : "0x0008588100347202397113858829294402397113858913304260239711385896200782023971138591675765802397113858951474220239711385911659260239711385903210480023971138588669441202397113859171060300239711385910794906023971 3" }{ "_id" : ObjectId("52378b196058a7c73e23a39c"), "AA885054905220072C" : "0x0008588541734201887084858998974600188708485877473408018870848589711077801887084858842932060188708485890624718018870848587771639001887084858859788520188708485876773814018870848591717333801887084859095514120188708 4" }{ "_id" : ObjectId("52378b196058a7c73e23a3a2"), "AA885054905220072C" : "0x0008590083252401827081859164787000182708185904202382018270818589222073601827081858775747180182708185905382444018270818589642994018270818588004263201827081859030830940182708185913368396018270818588099078201827081 4" }{ "_id" : ObjectId("52378b196058a7c73e23a3a4"), "transaction" : { "BIN" : "8588284676801797079", "date" : "2015/02/27 10:08:25 PM", "events" : [{ "date" : "2015/02/27 10:08:25 PM", "message" : "<message>", "type" : "ERROR" }, null], "imei" : "351777040332536", "readings" : [{ "date" : "2015/02/27 10:08:25 PM", "level" : "29299" }, null, { "date" : "2015/02/27 10:08:25 PM", "level" : "29299" }, { "date" : "2015/02/27 10:08:25 PM", "level" : "29299" }], "transactions" : [{ "date" : "2015/02/27 10:08:25 PM", "id" : "<id>", "odo" : "1000871", "uid" : "d7d3a7d49b8a42ef9fed018c0cce405d", "volume" : "4969" }, { "date" : "2015/02/27 10:08:25 PM", "id" : "<id>", "odo" : "1000871", "uid" : "d7d3a7d49b8a42ef9fed018c0cce405d", "volume" : "4969" }, { "date" : "2015/02/27 10:08:25 PM", "id" : "<id>", "odo" : "1000871", "uid" : "d7d3a7d49b8a42ef9fed018c0cce405d", "volume" : "4969" }], "version" : "0" } }{ "_id" : ObjectId("52378b196058a7c73e23a3b3"), "deviceinfo" : { "ISO" : "ISO7224023", "analog1" : "1", "analog2" : "0", "com1" : "1", "com2" : "1", "country" : "JAP", "date" : "2003/12/01 09:08:25 AM", "imei" : "357023006714589", "manufacturer" : "Not Known", "patent" : "PAT8015", "type" : "NOTI", "version" : "1" } }{ "_id" : ObjectId("52378b196058a7c73e23a3c5"), "transaction" : { "BIN" : "8588284676801407057", "date" : "2003/09/26 06:14:25 PM", "events" : [{ "date" : "2003/09/26 06:14:25 PM", "message" : "<message>", "type" : "ERROR" }, null], "imei" : "357541000234567", "readings" : [{ "date" : "2003/09/26 06:14:25 PM", "level" : "49875" }, null, { "date" : "2003/09/26 06:14:25 PM", "level" : "49875" }, { "date" : "2003/09/26 06:14:25 PM", "level" : "49875" }], "transactions" : [{ "date" : "2003/09/26 06:14:25 PM", "id" : "<id>", "odo" : "1004918", "uid" : "7b87089037894ab98296d9d8aa7560b7", "volume" : "7117" }, { "date" : "2003/09/26 06:14:25 PM", "id" : "<id>", "odo" : "1004918", "uid" : "7b87089037894ab98296d9d8aa7560b7", "volume" : "7117" }, { "date" : "2003/09/26 06:14:25 PM", "id" : "<id>", "odo" : "1004918", "uid" : "7b87089037894ab98296d9d8aa7560b7", "volume" : "7117" }], "version" : "3" } }{ "_id" : ObjectId("52378b196058a7c73e23a3c7"), "deviceinfo" : { "ISO" : "ISO7224023", "analog1" : "0", "analog2" : "0", "com1" : "0", "com2" : "1", "country" : "USA", "date" : "2005/01/25 06:59:25 PM", "imei" : "357541000234567", "manufacturer" : "Not Known", "patent" : "PAT2683", "type" : "ERROR", "version" : "3" } }{ "_id" : ObjectId("52378b196058a7c73e23a3cb"), "transaction" : { "BHEX" : "0x0008587733861801337053858863060500133705385891310212013370 3", "date" : "2003/08/30 07:26:25 AM", "id" : "<id>", "imei" : "357541000234567", "odometer" : "<odometer>", "uid" : "85c5b0e46ad84037bcb1ad29b08d59b5", "version" : "3", "volume" : "2595" } }{ "_id" : ObjectId("52378b196058a7c73e23a3d8"), "status" : { "analog1" : "1", "analog2" : "0", "com1" : "0", "com2" : "1", "date" : "2015/08/26 12:57:25 AM", "imei" : "357023006714589", "status" : "ER4", "type" : "NOTI" } }

APPENDIX B: SITUATION INTERVIEW WITH NFM

Interviewer: Interview between Mr Francis Oosthuysen and Romeo Botes, the researcher. Date and time 10 September 2013 at nine o'clock. Location Northwest University, Vaal Triangle Campus, building 8, room G05, Vanderbijlpark. Welcome, Mr. Francis Oosthuysen to the interview. I would like to thank you for this opportunity.

Interviewee: Thank you.

Interviewer: Just to introduce myself at this moment, I'm Mr. A R Botes, I'm doing currently my Masters in Computer Science at the Northwest University, and my study is a study I'm doing on unstructured document data stores. That is one of the data models available in one of the new emerging technologies known as NoSQL database management systems. Really, the idea of my study is to build an algorithm that could analyse this data that is unstructured and present some structured output for analysis and interpretations. The purpose of this interview is to find out whether your company may contain unstructured data, whether the company may benefit from this unstructured data and whether application aids to identify this unstructured data, that would aid in analysis. The data collected will be subject to qualitative data analysis from which conclusions will be drawn. Do you allow this study to use this data in the interview and publish if needed?

Interviewee: 0:01:40.5 Yes, that's fine.

Interviewer: Okay. So let's start with the interview. Okay, just a few warm-up questions to make you feel a little bit more relaxed. In what industry does the company resort?

Interviewee: 0:01:54.6 The company, Newcom Management resorts in quite a couple of- quite a few industries. We work in the agricultural industry, we work in the mining industry, the transport industry- the marine industry and the transport industry.

Interviewer: Okay, and what business does the company conduct?

Interviewee: 0:02:17.7 Primarily we manage the hydrocarbons used by the companies in the markets that I just mentioned.

Interviewer: Would you explain hydrocarbons, just briefly?

Interviewee: 0:02:29.7 Hydrocarbons primarily- on the hydrocarbons products we focus on diesel fuel and then oil products, lubricants, and what we basically do is we use- we consume large amounts of these valuable products in current market conditions. We put down electronic control systems to help them monitor and manage the consumption of these products.

Interviewer: Okay, thank you. How long has the company been in existence, or operation?

Interviewee: 0:03:02.2 Newcom has been in operation since 2011, so we are currently in our third year.

Interviewer: Was there any private business before 2011?

Interviewee: 0:03:12.5 Yes. Newcom was actually part of another company called Tecma Optimisation, and they started with the fuel management systems back in the 1990s, and the vision in the company outgrew the company, and it was a well thought out plan that Newcom was established to take on this market on its own and expand the business of fuel management in the Southern African marketplace.

Interviewer: Okay, what is the customer base for the company?

Interviewee: 0:03:44.9 The customer base is basically any customer or client organisation which uses or has its own refuelling infrastructure, meaning its own bulk storage tanks of fuels and pumping infrastructure, and examples is in the agricultural industry where you have large commercial farmers. They carry their own diesel fuel onsite and their own pumping infrastructure. And the transport industry, people and companies who has their own fuel- fleets, who carry their own infrastructures, and the mining operations who have their own refuelling infrastructure onsite and carry all their own bulk supply onsite.

Interviewer: Okay, where does the company resort [sic]?

Interviewee: 0:04:29.8 We have two offices we operate from. The one is in Cape Town, Durbanville. That's our administrative head office, and the operational head office is based in Boshoff in the Free State.

Interviewer: Okay. Where does the customer base of the company resort?

Interviewee: 0:04:48.7 Currently we are servicing clients across South Africa. We have clients in the Western Cape, Eastern Cape, Northern Cape, Free State, Gauteng, Limpopo, Mpumalanga and KwaZulu Natal, so we just about work in all the provinces. And we're now moving into Africa as well. So we have a broad range of clients in a diverse marketplace with diverse needs located in diverse geographical locations.

Interviewer: Thank you for that information about the business. Just a bit of introductory- Okay, so now we're going to start with the main interview questions. What data overall does the company collect?

Interviewee: 0:05:30.2 We collect primarily fuel-related or fluid-related data, and that consists of the fuel used by a vehicle or a person, the production from that fuel obtained, typically locations, where fuel was dispensed or used, times and places, the reasons, and I think that's primarily the data set that we have.

Interviewer: Okay. How is this data collected?

Interviewee: 0:06:04.5 At this point we have electronic control systems and it is a combination of electronic components, typically items like RFID technology which is allocated to users of the system, vehicles or people which identify themselves electronically, and then there's also manual input facilities put into keypads on the industrial computers onsite, where they punch in information, and that is typically kilometres of vehicles between fill-ups or hours of machines, and ja, that's primarily that.

Interviewer: Okay. Some of these devices or electronic devices you're speaking about, are they built by you or custom built by third parties?

Interviewee: 0:06:53.8 We have ventured to build our own equipment and it is currently being tested, and then we also use a third party company who manufactures equipment for us.

Interviewer: Okay. The methods used for the data collection, do they differ?

Interviewee: 0:07:09.8 They do. We have various methods that we currently employ, and it might probably evolve in the future as well, as technology goes, because we are focused on technology innovation. We currently use mechanisms- we just discussed that we capture data on the field by manual input or by electronic

device or other methods, and how we communicate the data eventually to our main data collection points. It can be via a Wi-Fi network, it can be via a fixed LAN network, it can be via GPRS-type connections, it can be via satellite connections, it can be via radio-type links. In the past it has been done via telephone modem communications, and then we have a fixed- we can do a fixed site localised installation where we put down service onsite as well, then it's a closed-loop system like that. So we have a very diverse system configuration and we work within all these configurations and connect within all these configurations' data. We manage it accordingly.

Interviewer: Okay, with these different methods of data collection I must assume that the data differs. Could you please explain how this data differs?

Interviewee: 0:08:28.3 Yes. Obviously the data will differ because we have different clients and different markets with different needs, whereby we capture different data for different clients and communicate them in different mechanisms, and also the mediums available. For instance, if I have a hardwire system or a fibre-connected system versus a to-go [sic] system it makes transporting or transmitting data a lot more easy for us, and thereby can access a lot more data, whereby I need to reduce my workload on the GPRS connection line. So yes, data differs in terms of input, and then also because of functionality and client dependence.

Interviewer: Thank you. For what is the data used that is collected?

Interviewee: 0:09:14.8 Currently it's to give our clients a competitive edge, primarily. To elaborate on that, it means that fuel is an important input cost in their industry, and for them to remain competitive in their market, their respective markets, they need to use their resources a lot more intelligently and a lot more optimally, and we come in and give them a system which can analyse the usage of their fuels in their vehicles, and report to them in real time what they are doing, and we also try and advise to them what they should be doing to eventually give them a competitive edge and help them reduce costs and achieve savings.

Interviewer: Okay, if I may ask, because there's a lot of different methods and a lot of different data that's being collected, in what type of structure or format does this data arrive at you?

Interviewee: 0:10:14.3 At this point the information on the field side from the raw industrial computers is in a raw text file with a fixed structure, which is then converted into a MySQL database. That's the fixed method at this time.

Interviewer: Okay. And what is included in this data?

Interviewee: 0:10:39.1 Basically there's a lot of open- there's a lot of space for data, and the primary set that we look at is date and time, fuel quantities, reasons, people, users, locations, errors, abnormalities, and then probably a couple of other variables.

Interviewer: Okay, you said that the data is stored in MySQL. Could you just please elaborate a little bit more on that. Is that structured data, very structured?

Interviewee: 0:11:18.2 Yes. It's always structured on a back-office site, a hex-service on the MySQL database, which is a hex-structured data set that can be- So all our different equipment from the different device groups have a fixed structure for us to be able to work with that data.

Interviewer: Okay, what protocols do you- Sorry, this data is now sent from these devices, and these devices obviously needs to store this now in a very structured database. Is there some kind of protocol or things that is used to transfer or analyse or whatever the data?

Interviewee: 0:12:06.7 Ja, that's a difficult one, because what we- the database is a fixed-structure set, and all the equipment that we now use needs to conform with the record that we obtain from the field device, needs to conform to that fixed structure. So yes, there is a lot of time and money spent on developing the protocol for each field type of us, and it necessarily means that we need to go to a development cycle on the software side when we include new hardware, because it's a- we need to integrate a new device with the new protocol to eventually get the data into our fixed database structure.

Interviewer: Okay, from the fixed database structure you mentioned, then consumers or customers would like to read the data. In what method do they read the data, spreadsheets, or do they receive e-mails? What format?

Interviewee: 0:13:02.1 We at this point use primarily a web-based reporting platform whereby web pages regroup the data and show that to them, to eventually give them information, and we do use a form of reporting in Excel-based reports, and then also

we have some sms and e-mail alarm-type information that we also send. But it would be nice to conform to the trends in the marketplace, so to have a look at the current social media platforms that's available and trying to use them in the future would also be a very strong point for us.

Interviewer: Okay, thank you. Is all the data captured and utilised by these devices?

Interviewee: 0:13:55.6 No, I don't think so. I think there's most probably a lot more data that can be captured.

Interviewer: Could you please explain why?

Interviewee: 0:14:06.2 We are limited to the structure within a group, the equipment itself and of the database, so we can only capture the parameters within that structure.

Interviewer: Okay. Do you suspect- or is there a possibility for hidden data from these devices?

Interviewee: 0:14:27.1 I believe so. I believe so, definitely.

Interviewer: Could you explain why?

Interviewee: 0:14:30.9 I think there is a lot of complex relationships that can be derived from having a more elaborate data structure and looking at the relationships between different parameters. At this point, because we do not have all the parameters, or even more parameters, it's difficult to ascertain what relationships can exist or cannot exist.

Interviewer: What data do you suspect from this then- or what type of information do you suspect?

Interviewee: 0:15:05.6 I think if there's additional parameters it might be interesting to note, specifically in our applications, to see if we have trends of vehicle[s] who has high fuel usages, the basis of some other form of operation that they are currently doing, which parameter we do not capture currently. So if we can expand the data selection and capture more parameters we can actually deduce a lot more information on why they used the amounts and how they are using them, a lot more- we can derive a lot more accurate estimation of what's going on.

Interviewer: Does the company have a program or software or application that analyses the current data onto real dissemination [sic] of information or is it just discarded?

Interviewee: 0:16:00.7 Ja, at this point it's just discarded. We do not encapsulate [sic].

Interviewer: Okay, so you take the data currently as it is?

Interviewee: 0:16:09.0 We just take the data as it is.

Interviewer: What benefit could the company gain from this unused data?

Interviewee: 0:16:15.0 Competitive edge. We can develop new algorithms for clients based on data captured that can predict a lot more accurately certain situations in terms of fuel usage, fuel monitoring, equipment, production, statistics, and we can actually add a lot more value to them by capturing a lot more information and giving them some feedback which was never even looked at.

Interviewer: How would the company benefit in the future by storing the data in an unstructured and raw data format?

Interviewee: 0:16:58.3 I think it can. First of all it can open up the possibility of acquiring a lot more types of equipment, physical hardware that can be used, and also reducing the development time of the protocol integration and implementation. So effectively lowering input cost, then deriving a lot more complex information and reporting information to clients can add value to them and increase revenue for us. And then obviously breaking new ground in our market can improve technology and create new opportunities which we are not even now aware of, so new business development is also a big possibility.

Interviewer: Possible business development, that sounds like a massive advancement for a company?

Interviewee: 0:17:46.4 Yes.

Interviewer: Would it be possible for some of these data capturing methods, or protocols or drivers, to be changed to accommodate a new database management technology?

Interviewee: 0:17:57.5 I think so, but that's a question for the developers to answer.

Interviewer: Okay, could it be possible for the company to provide some existing data for analysis?

Interviewee: 0:18:09.9 Yes, it is possible.

Interviewer: Okay, in this study, is this study allowed to use this data subjectively to find information among...

Interviewee: 0:18:21.2 Yes, it is.

Interviewer: Thank you very much. When will you know that this analysis of the data is correct?

Interviewee: 0:18:32.4 Hm... That is also a question for the developer.

Interviewer: Also a question for the developer?

Interviewee: 0:18:41.5 Yes.

Interviewer: Okay, isn't there like attributes to which you can compare maybe, like existing attributes?

Interviewee: 0:18:47.8 Yes. Well, the existing structures, what we have, so that's what I can use to measure existing attributes, but new attributes, it's going to be new territory for us as well.

Interviewer: Okay, in what way will this hidden data help you achieve your goals in the company?

Interviewee: 0:19:06.6 Well, the goal of the company is to make money, and if I can add more value with my system than the competition system can, then I will get the business and I can sustain the business by continuously giving my client valuable information, and I can only do that by capturing more data and deriving more relationships from what we currently have.

Interviewer: What are your expectations from the data?

Interviewee: 0:19:35.3 The expectations are to- with the- I would like to have the idea that with the same system, using it a little bit more intelligently, I can give a lot more value at the end of the day to my client.

Interviewer: Okay, would you even be willing to pay for this data analysis?

Interviewee: 0:19:54.1 I think it is a most valuable service, yes.

Interviewer: Okay, is there any information that you would like to add?

Interviewee: 0:20:04.4 At this point I don't think so, just that to think of a possibility that I can capture data from devices, and somehow get them into the same database for analysis, and also on structured data in the sense that I can capture more than I think I could have captured, it can add so much value. It can open up the future for a lot of different industries, so it's a very, very innovative research topic for me.

Interviewer: Okay, is there anything that you would like to know more about?

Interviewee: 0:20:50.0 Yes, how it will work.

Interviewer: How it will work. Okay, that's kind of a good question for me. Well, basically what's going to happen is as the data is stored in an unstructured database, hopefully with your developer now changing a little bit codes here to modify, to take it to the new database, then what the application should do, it should run through all the records within the database that gets now dumped into the database, and then try to identify all the attributes and possible relationships between these attributes, data and even possibly identify entities that could aid in the data analysis, and thereby you could eliminate existing information or existing data and attributes, and then conclude that there are new information, or information you have not known about.

Interviewee: 0:21:47.5 So would it be then possible in the future to expand on this relationship ability, to- The more data I'm getting the more possibilities there are for new relationships, stuff like that?

Interviewer: Yes. I think it falls more towards a BI initiative.

Interviewee: 0:22:07.0 Ah.

Interviewer: Business Intelligence.

Interviewee: 0:22:08.9 Yes

Interviewer: The goal of the study is not to go yet to business intelligence but it could look at the further possibility...

Interviewee: 0:22:17.0 Ja.

Interviewer: ... into that, or we only at this minute want to take unstructured data and try to put it in a structured format for people to analyse and see what could happen with the data, or what are they missing out on.

Interviewee: 0:22:30.4 All right. All right.

Interviewer: Any other questions?

Interviewee: 0:22:33.2 What IT infrastructure would be required on my side?

Interviewer: There won't be required much, any real IT infrastructure. The basic setup is the device will still have to report to an application server, kind of, because that's what I could use at this moment, and then you've got your normal database server.

Interviewee: 0:22:57.8 Mm hm.

Interviewer: So from the requirements, IT perspective, maybe an additional database server, but you could host now the new unstructured data on that, and then maybe a new application server that would analyse or process the data...

Interviewee: 0:23:14.6 Mm hm.

Interviewer: ... to reveal the new meaning or find the new attributes within the data.

Interviewee: 0:23:18.7 One of the questions I also had was, now that you're talking about it- and I mentioned this earlier- was communication quality is a big problem in South Africa, in Africa, so in my mind now an unstructured record necessarily means that in certain instances there will be a reduced record, information, and that might transmit a little bit more efficiently than a fixed structured type of thing.

Interviewer: Yes, that will definitely happen with unstructured data because you don't have to worry anymore about having a certain fixed length of the data. It will vary now according to device, so every device will- I don't know, you will have to give more details on that, right, you will know more about it on the devices.

Interviewee: 0:24:04.9 Ja.

Interviewer: But from my perspective it won't really matter how long the strings are, they will just get dumped into the database because the database will basically accept any data form that it can take.

Interviewee: 0:24:17.9 I'll work on it.

Interviewer: Excellent. And Jerry, I would just like to thank you for this interview and the opportunity for having this interview from you. Thank you. In this- I will now conclude the interview and stop the recording. Thank you.

APPENDIX C: COMPLETE PSEUDOCODE FOR INITIAL STUDY

```
1   class cls_Entity
2           String var_name
3           List lst_Attributes
4   end class
5
6   class cls_Attribute
7           String var_name
8           List lst_Values
9   end class
10
11  class cls_Value
12          String var_content
13          Integer var_count
14  end class
15
16  class cls_Relationship
17          String var_Entity_one
18          String var_Entity_many
19  end class
20
21  class cls_Document_store_analyser  // modified
22          List lst_Entities
23          List lst_Relationships
24
25      function add_entity(name)
26              if lst_Entities.contains(name)
27                  // do nothing
28              else
29                  cls_Entity obj_newEntity  = new cls_Entity()
30                  obj_newEntity.var_name  = name
31                  obj_newEntity.lst_Attributes  = new List
32                  lst_Entities.add(obj_newEntity)
33              end if
34      end function
35
36      function add_entity_attribute(name,  attribute, value)
37              cls_Entity obj_currentEntity  = lst_Entities(name)
38              if obj_currentEntity.lst_Attributes.contains(attribute)
39                  cls_Attribute obj_currentAttribute  = obj_currentEntity.lst_Attributes(attribute)
```

```
40                  if currentAttribute.values.contains(value)
41                      obj_currentAttribute.lst_Values(value).var_count =
42                          obj_currentAttribute.lst_Values(value).var_count + 1
43                      obj_currentEntity.lst_Attributes(attribute) = obj_currentAttribute
44                  else
45                      cls_Value obj_currentValue = new cls_Value()
46                      obj_currentValue.value = value
47                      obj_currentEntity.lst_Attributes(attribute).lst_Values.add(obj_currentValue)
48                  end if
49              else
50                  cls_Attribute obj_newAttribute = new cls_Attribute()
51                  obj_newAttribute.var_name = attribute
52                  cls_Value obj_newValues = new cls_Value()
53                  obj_newValues.var_content = value
54                  obj_newValues.var_count = 1
55                  obj_newAttribute.lst_Values = obj_newValues
56                  obj_currentEntity.lst_Attributes.add(obj_newAttribute)
57              end if
58      end function
59
60      function add_relationship(entity_one, entity_many)
61              if lst_Relationships.contains(entity_one + entity_many)
62                  // do nothing
63              else
64                  cls_Relationship obj_newRelationship = new cls_Relationship()
65                  obj_newRelationship.var_Entity_one = entity_one
66                  obj_newRelationship.var_Entity_many = entity_many
67                  lst_Relationships.add(obj_newRelationship)
68              end if
69      end function
70
71      routine initiation() // modified
72              //connect to server
73              //get List of collections.
74              //loop thru collections
75              //loop thru documents in collection
76              for each doc in collection
77                  document_analysis(doc, collection.name)
78              loop
79      end routine
80
81      function document_analysis(doc, parent)
```

```
82              List lst_Elements = doc.elements
83              for each element in lst_Elements
84                      if element.istypeDocument
85                          add_entity(element.name)
86                          add_relationship(parent, element.name)
87                          document_analysis(element.value, element.name)
88                      elseif element.istypeArray
89                          add_entity(element.name)
90                          add_relationship(parent, values.name)
91                          array_analysis(element.value, element.name)
92                      else
93                          add_entity_attribute(parent, element.name)
94                      end if
95              loop
96      end function
97
98      function array_analysis(arr, parent)
99              List lst_Values = arr.values
100             for each value in lst_Values
101                     if element.istypeDocument
102                         add_entity(values.name)
103                         add_relationship(parent, element.name)
104                         document_analysis(values.value, values.name)
105                     elseif element.istypeArray
106                         add_entity(values.name)
107                         add_relationship(parent, values.name)
108                         array_analysis(values.value, values.name)
109                     else
110                         add_entity_attribute(parent, values.name)
111                     end if
112             loop
113     end function
114
115     function output()
116         String var_output = "Entities and Attributes:" + NewLine
117
118             for each entity in lst_Entities
119             var_output = var_output + entity.name + "("
120
121                 if entity.Attribute.Count > 0
122                     for each attribute in entitly.lst_Attributes
```

```
123                         Integer var_value_count = 0
124                         for each value in attribute.lst_Values
125                             if var_value_count < value.var_count
126                                 var_value_count = value.var_count
127                             end if
128                         loop
129                         if var_value_count > 1
130                             var_output = var_output + attribute.var_name + ","
131                         else
132                             var_output = var_output + attribute.var_name + " (PK) ,"
133                         end if
134                     loop
135                 else
136                     var_output = var_output + ","
137                 end if
138                     //remove last 2 characters with subString from output and add NewLine
139             loop
140
141             var_output = "Relationships:" + NewLine
142
143             for each relationship in lst_Relationships
144                 var_output = var_output + relationship.var_Entity_one + "---------<" +
145                         relationship.var_Entity_many + NewLine
146             loop
147
148         return var_output
149
150     end function
151 end class
```

APPENDIX D: ACTUAL CODE FOR INITIAL STUDY

(Actual artefact programming code in Visual Basic .NET)

```
Imports MongoDB.Bson
Imports MongoDB.Driver
Imports MongoDB.Driver.Builders


Public Class cls_Document_store_analyser

    Public databasenames As List(Of String) ' temp list for db names
    Public collectionnames As List(Of String) ' temp list for collection names for selected databases

    Public lst_Entities As New SortedList
    Public lst_Relationships As New SortedList


    Public Sub Start(ByVal databasename As String, ByVal collectionname As String) 'Entry point for
    algorim, starting with the connection to mongo db.
        Console.WriteLine("Connect...")

        Dim mongo As MongoServer = MongoServer.Create("mongodb://localhost:27017")
        mongo.Connect()

        Console.WriteLine("Connected")
        Console.WriteLine()

        Dim db = mongo.GetDatabase(databasename)

        Using mongo.RequestStart(db)
            Dim collection = db.GetCollection(Of BsonDocument)(collectionname)

            For Each document As BsonDocument In collection.FindAll
                Try

                    Console.WriteLine("<<<<<<<<<<>>>>>>>>>>")
                    Add_Enity(document.Names(0))
                    Document_Analysis(document, 0, document.Names(0))

                Catch ex As Exception

                End Try

            Next
        End Using

        Console.WriteLine()
        Console.Read()
        Console.WriteLine("Disconnecting...")
        mongo.Disconnect()
        Console.WriteLine("Disconnected")
    End Sub


    Public Function Print_Output() As String

        Dim var_output As String = "Entities and Attributes:" & vbCrLf

        For i As Integer = 1 To lst_Entities.Count

            Dim obj_currentEntity As cls_entity = lst_Entities.GetByIndex(i - 1)

            var_output &= obj_currentEntity.var_name
            var_output &= " ( "

            If obj_currentEntity.lst_Attributes.Count > 0 Then

                For j As Integer = 1 To obj_currentEntity.lst_Attributes.Count
                    Dim att As cls_Attribute = obj_currentEntity.lst_Attributes.GetByIndex(j - 1)
```

```
                Dim count As Integer = 0

                For k As Integer = 1 To att.lst_Values.Count
                    Dim value As cls_value = att.lst_Values.GetByIndex(k - 1)


                    If count < value.var_count Then
                        count = value.var_count
                    End If

                Next

                If count > 1 Then
                    var_output &= att.var_name & " , "
                Else

                    var_output &= att.var_name & " (PK) , "
                End If

            Next

        Else
            var_output &= " , "
        End If

        var_output = var_output.Substring(0, var_output.Length - 2) & ")" & vbCrLf

    Next

    var_output &= vbCrLf & "Relationships:" & vbCrLf

    For i As Integer = 1 To lst_Relationships.Count
         Dim relatinship As cls_relationship = lst_Relationships.GetByIndex(i - 1)
         var_output &= relatinship.var_Entity_one & "-------<" & relatinship.var_Entity_many &
         vbCrLf
    Next

    Return var_output

End Function

Public Sub add_relationship(ByVal entity_one As String, ByVal entity_many As String) ' add
relationship between entity one and entity 2 based on entity many have multiple orcurences in
entity one


    If entity_one = "_id" Then
        entity_one = currentcollection
    End If

    Dim obj_newRelationship As New cls_relationship
    obj_newRelationship.var_Entity_one = entity_one
    obj_newRelationship.var_Entity_many = entity_many

    If lst_Relationships.ContainsKey(entity_one & "-----<" & entity_many) Then
        'Do Nothing
    Else
        lst_Relationships.Add(entity_one & "-----<" & entity_many, obj_newRelationship)
    End If



End Sub

Public Sub add_enity(ByVal name As String) ' add entity


    If name = "_id" Then
        name = currentcollection
    End If

    If lst_Entities.ContainsKey(name) Then
        'Do Nothing
    Else
```

```
        Dim obj_newEntity As New cls_entity
        obj_newEntity.var_name = name
        obj_newEntity.lst_Attributes = New SortedList
        lst_Entities.Add(name, obj_newEntity)

    End If


End Sub



Public Sub add_enity_attribute(ByVal name As String, ByVal attribute As String, ByVal value As
String) ' update an entity by adding an atribute to that entity

    Try

        If name = "_id" Then
            name = currentcollection
        End If

        Dim obj_currentEntity As cls_entity = lst_Entities.Item(name)

        If obj_currentEntity.lst_Attributes.ContainsKey(attribute) Then

            Dim obj_currentAttribute As cls_Attribute =
            obj_currentEntity.lst_Attributes(attribute)

            If obj_currentAttribute.lst_Values.ContainsKey(value) Then

                Dim obj_currentValue As cls_value = obj_currentAttribute.lst_Values(value)

                obj_currentValue.var_count += 1

                obj_currentAttribute.lst_Values(value) = obj_currentValue

            Else

                Dim newvalue As New cls_value
                newvalue.var_content = value
                newvalue.var_count = 1
                obj_currentAttribute.lst_Values.Add(value, newvalue)

            End If

        Else

            Dim obj_newAttribute As New cls_Attribute
            obj_newAttribute.var_name = attribute
            obj_newAttribute.lst_Values = New SortedList

            Dim obj_newValue As New cls_value
            obj_newValue.var_content = value
            obj_newValue.var_count = 1

            obj_newAttribute.lst_Values.Add(value, obj_newValue)

            obj_currentEntity.lst_Attributes.Add(attribute, obj_newAttribute)

        End If

        lst_Entities.Item(name) = obj_currentEntity

    Catch ex As Exception

    End Try

End Sub

Public Sub Document_Analysis(ByVal doc As BsonDocument, ByVal depth As Integer, ByVal ent As
String) ' algorithm for processing BSON Documents
```

```
    Dim elements As System.Collections.Generic.List(Of BsonElement) = doc.Elements

    Add_Enity(ent)

    For Each element As BsonElement In elements

        If element.Value.IsBsonDocument Then ' Check if element in document is a BSON document if
        so, add it as entity and then call BDoc recursively.

            Try

                If element.Name.Length > 0 Then ' in some instance the document object just not
                contain a name, but still want to store the values
                    add_enity(element.Name)
                    add_relationship(ent, element.Name)
                    Document_Analysis(element.Value, depth + 1, element.Name)
                    Console.WriteLine(depth & " Document>>>> " & element.Name)
                Else ' in some instance the array object just not contain a name, but still want to
                store the values
                    add_enity(ent)
                    Document_Analysis(element.Value, depth + 1, ent)
                    Console.WriteLine(depth & " Document>>>> " & "No Name")
                End If

            Catch ex As Exception

            End Try

        ElseIf element.Value.IsBsonArray Then ' Check if element in document is a array if so, add
        it as entity and then call BArray recursively.

            Try

                If element.Name.Length > 0 Then
                    add_enity(element.Name)

                    add_relationship(ent, element.Name)
                    Array_Analysis(element.Value, depth + 1, element.Name)
                    Console.WriteLine(depth & " Array>>>> " & element.Name)
                Else ' in some instance the array object just not contain a name, but still want to
                store the values
                    add_enity(ent)
                    Array_Analysis(element.Value, depth, ent)
                    Console.WriteLine(depth & " Array>>>> " & "No Name")

                End If


            Catch ex As Exception

            End Try

        Else 'if element is not of type array or of type doucment then assume element is an
        attribute.
            Try
                add_enity_attribute(ent, element.Name, element.Value)
            Catch ex As Exception
                add_enity_attribute(ent, element.Name, "No Value")
            End Try


            Console.WriteLine(depth & " Attribute>>>> " & element.Name)

        End If


    Next

End Sub

Public Sub Array_Analysis(ByVal arr As BsonArray, ByVal depth As Integer, ByVal ent As String) '
algorithm for processing BSON Documents Array
```

```
    Add_Enity(ent)


    For Each value As Object In arr ' loop thru all array objects

        Try
            If value.IsBsonDocument Then ' Check if element in document is a BSON document if so,
            add it as entity and then call BDoc recursively.

                Try


                    add_enity(value.Name)
                    add_relationship(ent, value.Name)
                    Document_Analysis(value, depth + 1, value.Name)

                    Console.WriteLine(depth & " Document>>>> " & value.Name)

                Catch ex As Exception

                    add_enity(ent)
                    Document_Analysis(value, depth + 1, ent)
                    Console.WriteLine(depth & " Document>>>> " & "No Name")

                End Try


            ElseIf value.IsBsonArray Then ' Check if element in document is a array if so, add it
            as entity and then call BArray recursively.

                Try
                    add_enity(value.Name)
                    Console.WriteLine(depth & " Array>>>> " & value.Name)
                    add_relationship(ent, value.Name)
                    Array_Analysis(value, depth, value.Name)

                Catch ex As Exception

                    add_enity(ent)
                    Console.WriteLine(depth & " Array>>>> " & "No Name")
                    Array_Analysis(value, depth, ent)

                End Try


            Else 'if element is not of type array or of type doucment then assume element is an
            attribute.

                Try
                    add_enity_attribute(ent, value.Name, value.value)
                Catch ex As Exception
                    add_enity_attribute(ent, value.Name, "No Value")
                End Try


                Console.WriteLine(depth & " Attribute>>>> " & value.Name)

            End If
        Catch ex As Exception

        End Try

    Next

End Sub


' code for interaction with MongoDB using the MongoDB Driver.

Public Sub Get_Databases() ' get a list of current databases on the server.

    Console.WriteLine("Connect...")
```

```
        Dim mongo As MongoServer = MongoServer.Create("mongodb://localhost:27017")
        mongo.Connect()

        Console.WriteLine("Connected")

        Try
            databasenames = mongo.GetDatabaseNames
        Catch ex As Exception

        End Try

        Console.WriteLine("Disonnecting")
        mongo.Disconnect()
        Console.WriteLine("Disconnected")


    End Sub


    Public Sub Get_Collections(ByVal database As String) ' gets a list of current collections in
    database

        Console.WriteLine("Connect...")

        Dim mongo As MongoServer = MongoServer.Create("mongodb://localhost:27017")
        mongo.Connect()

        Console.WriteLine("Connected")
        Console.WriteLine()

        Dim db = mongo.GetDatabase(database)

        collectionnames = db.GetCollectionNames

        Console.WriteLine("Disonnecting")
        mongo.Disconnect()
        Console.WriteLine("Disconnected")
    End Sub


    Dim currentcollection As String = ""


    Public Sub Start_All_Collections(ByVal databasename As String) ' loop thru all collections
    avaliable in the database

        For Each collectionname As String In collectionnames
            currentcollection = collectionname
            Start(databasename, collectionname)
        Next

    End Sub


End Class
```

APPENDIX E: MONGODB SETUP AND DATA LOADING

APPENDIX E: MONGODB SETUP AND DATA LOADING

The following steps outline the installation process and loading of data using MongoDB.

STEP 1: Download MongoDB from:

http://downloads.mongodb.org/win32/mongodb-win32-x86_64-2.4.6.zip

STEP 2: Extract the archive to a folder on the computer.

Instance of this study: c:/mdb

STEP 3: Create a data folder

Instance of this study: c:/data

STEP 4: Open *Command Prompt* and navigate to extrated bin folder of MongoDB.

Instance of this study:

>>cd c:\mdb\bin

STEP 5: In *Command Prompt*, run MongoDB by typing “mongod”.

Instance of this study:

>>c:\mdb\bin>mongod

MongoDB should start up.

STEP 6: Open a second *Command Prompt* and navigate to extrated bin folder of MongoDB.

Instance of this study:

>>cd c:\mdb\bin

STEP 7: In *Command Prompt* run the following command to import the data from a JSON file.

Instance of this study:

>>c:\mdb\bin>mongoimport --db rae --collection customer --file rawdata.json

STEP 7 command terms decomposition:

Mongoimport : import program

--db rae : parameter specifying the database to store the import data in (rae).

--collection customer : parameter specifying the collection to store the imported data in (customer).

--file rawdata.json : parameter specifying the file location of the file that needs to be imported (rawdata.json).

STEP 8: Repeat STEP 7 for all other data that needs to be imported.

APPENDIX F: EVALUATION INTERVIEW WITH NFM

Interviewer: Evening, this is a follow up interview on the first interview, and that's with Mr. Francois Oosthuisen. The date and time is 6 November, 2013 at six o'clock. The location is 13 Kingfisher Mews, Sylviavale, Vanderbijlpark. Once again thank you for your time to do this interview.

Interviewee: 0:00:21.8 My pleasure.

Interviewer: 0:00:22.5 Great. The purpose of this interview is to find out- to evaluate the generated output of the artefact, given the data presented by the business, and to evaluate the artefact's utility within the RA, if you still at least remember what we spoke the last time about.

Interviewee: 0:00:40.6 Yes.

Interviewer: 0:00:41.3 So you do. Okay. After the previous interview I took the information that was provided by your business and passed it through the artefact and an output was generated. Could I please ask you to evaluate the output? You may scribble on the notes, on the paper, as needed, and after evaluation I will ask a few questions about your observations of the data. Just yet again I need to ask, do you allow this study to be used as data, from this interview, and publish it if need be?

Interviewee: 0:01:16.4 Yes.

Interviewer: 0:01:17.0 Okay, thank you very much. Before you start evaluating the data I would just like you to take into consideration the following questions of what existing data have you encountered, and then I will also ask you to please mark these existing data with a blue pen. And then secondly I would also like you to observe, like, what new, unknown or discarded data have you discovered, and would you please mark this with a red pen? Okay, I'm going to stop the recording now for the amount of time while Mr. Francois Oosthuisen is evaluating the generated output.

[break]

Interviewer: 0:02:05.0 Okay, Mr. Francois, could you please indicate what existing data have you observed in the generated output?

Interviewee: 0:02:15.1 Certainly. I've actually marked quite a couple of parameters here, and just to name but a few there would be capacity of the consumer, ?consumer groups, consumer types, consumer ID's, current operating hours or current limits, the descriptions, kilometre limits. And on the events side I would say consumer ID's again, the container ID's, the message, event names, descriptions. On the groups there's a consumer group ID, a description, the max kilometre- metres, name. On the readings there's the date and level. With the transactions it's the bypassed status, the consumer transaction, the consumer ID number, the CDK information, the dates, the kilometre per litre, locations, odometers and- to name but a few.

Interviewer: 0:03:09.0 Thank you for that, that's quite extensive. Could you please indicate or elaborate a little bit on what existing- or what new or unknown data, or discarded data have you now also encountered in the data presented, in the generated output presented?

Interviewee: 0:03:28.5 Certainly. I've actually marked them in red here as you suggested, and there is a couple of parameters which I don't really understand, which seemed to be still encoded, and then there's some parameters here which I've newly now discovered, and to name but a few again it's capture type, which was explained was the type of transaction. On the device side there's actually device status information which is also fairly new. On the transaction side there would be some remarks data, start and end time to source, and some version type parameters. That is all new to me at this point.

Interviewer: 0:04:09.2 Thank you very much. how would you rate the generated output in terms of the information that you would like- would have liked to gain?

Interviewee: 0:04:17.6 I think at this stage, looking at what there is and what- the potential, I would rate it very good at this point.

Interviewer: 0:04:23.8 Would you be able to use this data that was generated?

Interviewee: 0:04:26.9 Yes, certainly. Not only is there a lot of data captured that we currently use, there's also a lot of new data that's indicated that has been captured, which I think at this point can prove to be very valuable.

Interviewer: 0:04:41.6 Thank you. Could you please give an example?

Interviewee: 0:04:45.5 All right. Just looking at the start and end time parameters, which is new at this point, would be- a good example would be to look at the start and end time of a transaction and determine the time spent within a day in transactions to determine where production gets lost due to ineffective refuel operations, and this might be due to laziness or misuse, and that- certainly that can contribute greatly to the overall operations, and within my organisation.

Interviewer: 0:05:16.9 Thank you very much. What other examples could you maybe present us with, this new generated output?

Interviewee: 0:05:23.4 I think the capture type, which is a differentiation between the type of internal transaction, which is typically when they try and dispense this fuel on site, versus a transaction which is dispensed outside the site, to get the grasp of how many transactions is dispensed onsite and offsite versus litres, and actually see what the effectiveness of your current infrastructure is, that's ongoing. And then obviously the device status information is extremely powerful because you can do a lot of proactive maintenance by tracking and measuring your physical field hardware, and there find the stages, and in fact that your quality of signal coming through is still intact or up to specifications, so that's extremely valuable data.

Interviewer: 0:06:07.2 Super. Thank you very much. How would you rate the user ability of the output in terms of the information you would have liked to gain?

Interviewee: 0:06:15.7 Well, again, Sir, very good. There's a lot of extra value that's coming out, and incorporating this into a bigger system, a lot of additional value can be added to potential clients in the future.

Interviewer: 0:06:30.1 Thank you very much for all those examples. They will become quite handy. If I may ask, do you think that more possibilities are now possible with this new data that has been presented?

Interviewee: 0:06:44.2 I think so, and over time it's not just the new data, it's maybe the combination of new data and old data, deriving new combinations. That would also be a huge benefit because it's always trying to add more value with what you have, and looking differently at what we have can be a great, great asset to the organisation.

Interviewer: 0:07:06.2 Could you please elaborate a little bit more on the possibilities you think are available?

Interviewee: 0:07:10.7 Yes, certainly. Just looking at business environment currently, we need to optimise researchers, and being that the data is a huge resource for us to add value to clients, if I can optimise a current or existing resource to get more output from it, effectively, then that mean I can contribute to my bottom line of my company which makes me more sustainable, and if that means I do not have to incur new costs, I just have to look differently, or smartly-er at my existing data, then that's the future.

Interviewer: 0:07:40.0 Okay, what value does this provide the business?

Interviewee: 0:07:42.9 The value would be quantified in terms of financial gain, providing new and more value adding services to clients, effectively, and also sustainability. I mean, if we can put down systems which can do more than the competition, that means that we're going to continue to get new business and grow, and that feeds families and that creates job opportunities in the community, so it's a long way of- improving small stuff goes a long way, that people might neglect sometimes.

Interviewer: 0:08:15.8 Thank you very much. Would the business like to expand on this research incorporating the artefact into the business information structure?

Interviewee: 0:08:25.9 Ja, certainly. It makes an obvious- It's a no-brainer. It's an obvious choice for a business point of view to optimise your current resources, and yes, just looking at the current output I think there's such a big scope, so definitely.

Interviewer: 0:08:40.2 Thank you very much. This is basically the end of the main information, I'm just going to enter now the cool-off session whereby I'm going to ask you is there any information that you'd like to add?

Interviewee: 0:08:52.1 At this point it's difficult. There's a lot of information that's new, and it would take time to process the information and actually verify the validity and- but just looking at first glance there's a lot of value, and I guess that's probably the name of the game. You know, it's continuous improvement, it's continuous growth, it's continuous evaluation, and you're only in front as long as can

run the fastest, so I think at this point this is one of the paths that we would like to venture [into] in the future to try and optimise our business as well.

Interviewer: 0:09:28.6 Thank you very much. Is there anything that you would like to know more about?

Interviewee: It's still a complex subject and I'm trying to understand the physical research itself, I guess I'm not the subject expert. So ja, I think the possibilities in the future regarding multiple sets of hardware that can feed information to a centralised system in no specific format or structure, and then one centralised system which can decode and add value, I think the possibilities, it's quite tremendous. And I would just like to know physically, physical viability or realistic viability. Is that- how long would that be in the future for something like that if you have multiple sets of hardware platforms which communicate to a centralised data platform, and you have an artefact in this case which is effective in decoding multiple sets of data?

Interviewer: 0:10:27.3 Could you just repeat a question you would like to know, like...

Interviewee: 0:10:30.8 The timeframe, is the research in such a form that it can be transferred quickly into a physical product or system, or do you still think there's still some time left for research?

Interviewer: 0:10:41.5 Well, there is still time left for research, as always, but physically for us to- the artefact is representing a very simple algorithm code at this moment that can be used to incorporate into any language, almost, and by incorporating that into any language, that if the business is- makes use of a language that's easy for them to use and they can just incorporate the algorithm, then they could gain fast time rather than having the artefact in a very fixed structure and then having to try and interact with the artefact, in that manner. That's why the artefact was initially developed in an algorithm pseudocode kind of way, for more towards understandability rather than implementation. So if you really want to implement the artefact you can just take it and see how it works, and then just get your coders or developers to code it similarly to what you need.

Interviewee: 0:11:38.6 So if you can now try and explain it to a normal guy, how would this then fit into a physical system? Say we're doing our- I have multiple

sets of hardware in the field and I have my main database where my reports are generated from, and I'm going to put in this there now, this layer which is this artefact which is going to eliminate all my drivers in the future?

Interviewer: 0:12:01.0 Okay, well, how...

Interviewer: 0:12:02.2 Is that where it's going to fit in, is that the idea?

Interviewee: 0:12:04.9 Okay. Well, basically from your standpoint of where you capture the data from, the devices...

Interviewer: 0:12:10.0 Yes.

Interviewee: 0:12:10.3 Which will now obviously, instead of going to your structured format, going into now an unstructured format. Okay, so that's the first eliminations, where those drivers of trying to force it now into structured formats, that's already eliminated.

Interviewer: 0:12:22.3 All right.

Interviewee: 0:12:22.6 And once you've now got it in that unstructured format you can start doing processing, the normal processing, and converting the data into a into a structured format that you can use for your imports, etc. and what- where the artefact now comes in is like when you add a new device or existing devices and there's- like as you have shown- information that you did not even know about...

Interviewer: 0:12:44.8 Mm hm.

Interviewee: 0:12:44.9 The artefact should highlight this new information, and then automatically you will be able to incorporate that information into the new system. An expansion of future possibilities, actually automatically taking the data- now, just not showing the new data but automatically taking the data into new- into the structured database, already making the structured database really adaptable...

Interviewer: 0:13:08.5 Mm hm.

Interviewee: 0:13:08.8 ... to incorporate the new data. There's a concept known as attribute values, entity attribute value stores here in these- etc. It's a further concept, and also once again with a newer scale movement that came along where the relational database manages- is now catching up on horizontal scaleability,

where you want to have multiple computers, and data shared on these computers. Eventually I would recommend that. Like unstructured data would like to go into a structured format once again, having the structured format database as having these capabilities.

Interviewer: 0:13:45.6 So just clarify something, the artefact would have the ability to highlight newly identified...

Interviewee: 0:13:52.8 The existing artefact, yes, will highlight new...

Interviewer: 0:13:56.0 Because that would be a difficult task, to try and identify all the new stuff...

Interviewee: 0:14:00.8 Yes.

Interviewer: 0:14:01.0 So if that's done automatically that's also a major benefit.

Interviewee: 0:14:04.3 That's the whole idea. That's what the artefacts do, is present the new information, compared to existing information.

Interviewer: 0:14:10.8 All right, that's wonderful. It's extremely exciting.

Interviewee: 0:14:13.8 Okay, thank you very much for the interview...

Interviewer: All right, so thanks...

Interviewee: 0:14:17.2 Francois Oosthuisen.

Interviewer: 0:14:18.2 Thank you Mr. Romeo.

Interviewee: 0:14:21.8 I'm stopping the recording now.

APPENDIX G: CONSENT FORM

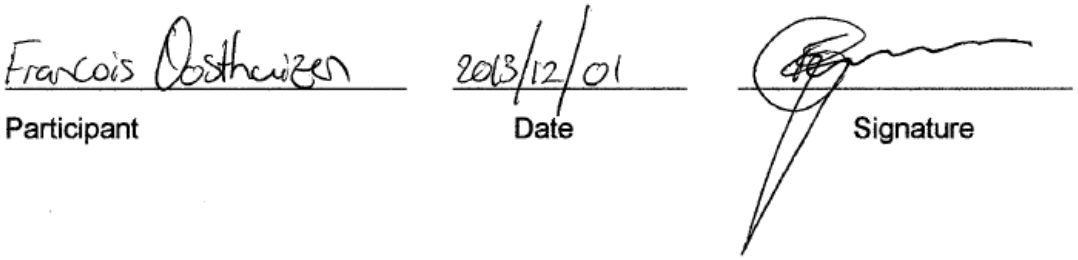
TITLE:

An artefact to analyse unstructured document data stores.

DETAILS OF THE RESEARCHER:

AR Botes
Junior Lecturer
School of Information Technology
NWU VTC
Vanderbijlpark

		Initial
1.	I confirm that I have understood the information provided for the above study and have had the opportunity to ask questions.	
2.	I understand that my participation is voluntary and that I am free to withdraw at any time, without giving reason.	
3.	I agree to take part in the above study.	
4.	I agree to the interview being audio recorded.	
5.	I agree to the use of quotes in publications.	



Participant	Date	Signature
Francois Oosthuizen	2013/12/01	