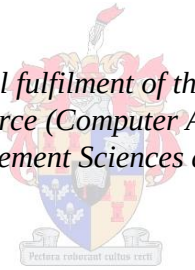
NoSQL database considerations and implications for businesses

by
Dawid Johannes Pretorius

Thesis presented in partial fulfilment of the requirements for the degree of Masters of Commerce (Computer Auditing) in the Faculty of Economic and Management Sciences at Stellenbosch University



Supervisor: Mr. Leon Pieter Steenkamp

December 2013

Declaration

By submitting this thesis electronically, I declare that the entirety of the work contained therein is my own, original work, that I am the sole author thereof (save to the extent explicitly otherwise stated), that reproduction and publication thereof by Stellenbosch University will not infringe any third party rights and that I have not previously in its entirety or in part submitted it for obtaining any qualification

December 2013

Abstract

NoSQL databases, a new way of storing and retrieving data, can provide businesses with many benefits, although they also pose many risks for businesses. The lack of knowledge among decision-makers of businesses regarding NoSQL databases can lead to risks left unaddressed and missed opportunities.

This study, by means of an extensive literature review, identifies the key drivers, characteristics and benefits of a NoSQL database, thereby providing a clear understanding of the subject. The business imperatives related to NoSQL databases are also identified and discussed. This can help businesses to determine whether a NoSQL database might be a viable solution, and to align business and information technology (IT) objectives.

The key strategic and operational IT risks are also identified and discussed, based on the literature review. This can help business to ensure that the risks related to the use of NoSQL databases are appropriately addressed. Lastly, the identified risks were mapped to the processes of COBIT (Control Objectives for Information and Related Technology) to inform a business of the highest risk areas and the associated focus areas.

Opsomming

NoSQL databasisse, 'n nuwe manier om data te stoor en herwin, het die potensiaal om baie voordele vir besighede in te hou, maar kan ook baie risiko's teweeg bring. Gebrekkige kennis onder besigheidsbesluitnemers oor NoSQL databasisse kan lei tot onaangespreekte risiko's en verlore geleenthede.

Hierdie studie, deur middel van 'n uitgebreide literatuuroorsig, identifiseer die sleutel eienskappe, kenmerke en voordele van 'n NoSQL databasis, om sodoende 'n duidelike begrip van die onderwerp te verkry. Die besigheidsimperatiewe wat verband hou met NoSQL databasisse is ook geïdentifiseer en bespreek. Dit kan besighede help om te bepaal of 'n NoSQL databasis 'n werkbare oplossing kan wees, asook sake- en inligtingstegnologie (IT) doelwitte in lyn met mekaar bring.

Na aanleiding van die literatuurstudie is die sleutel-strategiese en operasionele IT-risiko's geïdentifiseer en bespreek. Dit kan help om aan besighede sekerheid te verskaf dat die risiko's wat verband hou met die gebruik van NoSQL databasisse toepaslik aangespreek word. Laastens is die geïdentifiseerde risiko's gekoppel aan die prosesse van COBIT om 'n besigheid van die hoë-risiko areas en die gepaardgaande fokusareas in te lig.

CONTENTS

List of figures

Page

List of tables

Page

1. Introduction

1.1 Background

In 1970, Codd outlined a relational approach for databases in his paper “A relational model of data for large shared data banks” (Codd, 1970). Shortly thereafter, Structured English Query Language (SEQUEL; later renamed as Structured Query Language or SQL) (Chamberlin & Boyce, 1974), was presented to provide a way to access data in a relational database. The relational model has since become the dominant form in the database market (Indrawan-Santiago, 2012).

In September 2013 the most popular database management systems (DBMS) – by a large margin – were Oracle, Microsoft SQL server and MySQL (DB-Engines Ranking, 2013). All three DBMS have a relational database model and use SQL as query language. In light of its popularity, the relational database has proved to be an entrenched technology and a tried-and-tested database solution for businesses.

However, the limitations of the relational database model as a database solution for the many different business requirements became evident due to the increasing data and infrastructure needs of the large Web 2.0 companies (Leavitt, 2010). Mohan (2013) describes some of the reasons for the inadequacy of the relational database management system (RDBMS) as experienced by Web 2.0 companies as follows:

- The relational model was too rigid, and the modelling of Web 2.0 data could become problematic.
- RDBMS did not traditionally support schema evolution; therefore, it was considered as lacking the required flexibility.
- The programmers writing data manipulation code were required to learn SQL and to be an expert in another programming language.
- For certain application requirements, the RDBMS path lengths were deemed unacceptably long and the costs involved too high.
- RDBMS traditionally supported neither the scalability required by Web 2.0 companies nor the use of commodity hardware.
- Failure of individual nodes was not supported sufficiently for operations to continue smoothly.

- The data consistency requirements of the RDBMS were considered to be too stringent.

Amazon and Google, which were also faced with the limitations of the relational database, responded by designing their own database solutions. Both companies published a paper (DeCandia, Hastorun, Jampani, Kakulapati, Lakshman, Pilchin, Sivasubramanian, Vosshall & Vogels, 2007; Chang, Dean, Ghemawat, Hsieh, Wallach, Burrows, Chandra, Fikes & Gruber, 2008) that provided details of the design and of the implementation of their respective solutions that did not support the existence of a full relational model. This can be argued, was the trigger and the inspiration for the surge of NoSQL database solutions that came thereafter (Leavitt, 2010; Cattell, 2011; Mohan, 2013).

The characteristics of NoSQL databases are different from those of relational databases, although not every NoSQL database solution followed the same approach to storing and retrieving data. However, NoSQL databases are united in using a non-relational database model, or in other words, NoSQL databases do not use the relational database model (Leavitt, 2010).

In general, the characteristics of NoSQL can be defined as using a non-relational data model that is designed for distributed processing and horizontal scalability, and which allows for less strict rules regarding adherence to database schema and to the reduced consistency of data (Indrawan-Santiago, 2012). When compared to RDMS, NoSQL databases are still a relatively new concept that have not yet achieved widespread adoption, with most NoSQL vendors still being small start-ups (Adrian, 2012). Some of the challenges that are delaying the implementation of NoSQL databases as a mainstream solution are:

- Information technology (IT) decision-makers generally lack understanding of the strengths and weaknesses of NoSQL.
- Theoretical data modelling techniques are not available for assisting with the design of a database using different data models.

- The lack of a model with a strong theoretical foundation that explains how latency impacts the design and the performance of a database (Indrawan-Santiago, 2012).

Not everyone agrees that NoSQL is the future of databases (Stonebraker, 2011). Some argue that NoSQL should only be seen as an additional tool that can work together with relational databases to become a combined database solution (Lerner, 2010). However, large Web 2.0 companies have embraced the non-relational database approach.

1.2 Research problem and objective

NoSQL databases are still an immature technology and have not yet received mainstream adoption (Adrian, 2012). Due to the immaturity of the NoSQL database technology, a business that implements a NoSQL database solution will be faced with many significant business risks. The lack of understanding by IT decision-makers of the strengths and weaknesses of NoSQL database solutions (Indrawan-Santiago, 2012) will not only increase the risk of failure when implementing a NoSQL database, but also the risk that a business will miss opportunities related to the use of NoSQL databases.

The aim of this study is to identify the incremental risks related to businesses considering the use of a NoSQL database solution. The risks will be identified, and discussed, in such a way that it can assist a business to achieve improved alignment of their IT and business objectives and therefore also reduce the gap between IT and the business. The risks identified during this study will also be mapped to COBIT (Control Objectives for Information and Related Technology), a framework that assists businesses with the governance and management of IT, thereby enabling a business to better understand the impacts that these risks might have. By understanding the risks related to the use of NoSQL databases, these businesses will be better able to mitigate the relevant risks, and thereby ensuring greater success from the implementation thereof.

This study does not attempt to provide an overview of all the relevant risks that are related to the use of NoSQL databases. This study will only focus on the most significant incremental risks related to the use of NoSQL databases and will therefore exclude the generic risks that are related to the general use of databases. The generic risks that exist due to the general use of databases are omitted from this study, as these risks have already received extensive research that contributed to the entrenchment of the relational database technology. The incremental risks of NoSQL databases identified during this research assignment should therefore not serve as a generic comprehensive list, but should be regarded by a business as including specific business risks and the risks related to the specific NoSQL database product.

This study can also be used as a starting point for a business, in order to enable it to determine whether a NoSQL database is a viable option. This could also aid businesses that have already implemented a NoSQL database solution to better understand the risks that are related to using a NoSQL database.

1.3 Scope and limitations of the study

This study is focused on providing business managers with knowledge of NoSQL databases and of the business risks that they can introduce. Therefore, the study will not include technical discussions of the risks identified, or of the implementation guidelines.

1.4 Research methodology

This research consists of a non-empirical study that is based on an extensive literature review of aspects related to NoSQL databases. Based on the literature review, the key drivers, the characteristics, and the benefits of NoSQL databases are identified and discussed in Chapter 2.

Based on the characteristics identified in Chapter 2, the associated business imperatives are identified in Chapter 3. The business imperatives that are related to

NoSQL databases are useful for businesses to know, in order that they might improve on their alignment of the IT and business objectives involved.

In Chapters 4 and 5 the strategic risks and the operational risks are discussed respectively. Chapter 6 provides the mapping of these risks to COBIT, a business framework for the governance and the management of enterprise IT.

2. NoSQL: drivers, characteristics and benefits

2.1 Introduction

Eric Evans, a systems developer at Rackspace, is generally referred to as making the NoSQL term popular when Johan Oskarsson, an employee of Last.fm, organised an event in 2009 to discuss distributed structured data storage. The term, at first, was understood to be 'No to SQL', and received much criticism for dismissing the RDBMS and for sending an "inappropriate or inaccurate message" (Evans, 2009). However, the name was later changed, and is now referred to by most as 'Not only SQL' (Shalom, 2009b).

In section 2.2 of this chapter, the drivers of NoSQL will be discussed as a starting point. Thereafter, the main characteristics that NoSQL databases share will be discussed in section 2.3. Section 2.4 provides some key benefits of NoSQL as a database solution, and section 2.5 concludes the chapter with an overview of its contents.

2.2 Drivers of NoSQL

2.2.1 Web 2.0

With the emergence of such companies as Amazon, Google, Facebook, Twitter and LinkedIn, a dramatic increase in scale occurred on such dimensions as the numbers of users, and the amount of data collected and processed. With applications becoming more accessible over the Internet, the data created by the users has become increasingly integral to the value of Web 2.0 businesses (Couchbase, 2013). Web 2.0 can be defined as the second version of the World Wide Web, where web pages are not limited to passive viewing only, and therefore not static in nature. Web 2.0 websites enables users to interact and collaborate with each other (Wikipedia, 2013b).

The Web 2.0 businesses pushed the flexibility and the scalability of the relational database to its limits, with it becoming increasingly difficult to manage (Babcock, 2010; Indrawan-Santiago, 2012). The limitations of the RDBMS, as discovered by the large Web 2.0 companies, were the trigger for these companies to develop their own database solution, in the absence of commercial alternatives (Strauch, 2011; Couchbase, 2013; Mohan, 2013).

Amazon, a company that runs a worldwide e-commerce platform, developed Dynamo, describing it as a "highly available and scalable distributed data store" (DeCandia *et al*., 2007). Google developed Bigtable as a database solution for many of their products and projects, including web indexing, Google Earth, Google Analytics, Personalised Search, and Google Finance (Chang *et al*., 2008).

The new developments in data management devised by these companies, and the papers released by Amazon (DeCandia *et al.*, 2007) and Google (Chang *et al*., 2008) providing details of the design and implementation thereof, have sparked enormous interest, due to the increasing number of businesses experiencing the same relational database limitations. A huge number of mainly open-source NoSQL projects that are based on the developments of these Web 2.0 companies have since emerged (Hecht & Jablonski, 2011; Floratou, Teletia, DeWitt, Patel & Zhang, 2012; Couchbase, 2013). A list of over 140 NoSQL databases is given on the www.nosql-database.org website.

According to Brooks (2011), NoSQL has enabled web-based businesses with fast-growing user demands to exploit the huge quantity of data created by their users. Therefore, NoSQL databases are unquestionably one of the by-products that originated from the Web 2.0 era (Tudorica & Bucur, 2011).

2.2.2 Big data

According to a study performed by the International Data Corporation (Gantz & Reinsel, 2012), the amount of data that is available in the world is likely to grow from 130 exabytes in 2005 to 40 000 exabytes, or 40 trillion gigabytes, in 2020. The vast majority of new data being generated is classified as unstructured data, which can be described as data that are not organised into a well-defined schema.

Unstructured data include data types such as documents, email, multimedia and social media (Xiang, Hou & Zhou, 2010; Okman, Gal-Oz, Gonen, Gudes & Abramov, 2011). The data being collected are generated from an increasing amount of sources, including such data types as “personal user information, geo location data, social graphs, user-generated content, machine logging data, sensor-generated data”, and many more (Couchbase, 2013).

Businesses began to realise that big data, and the capture, integration, and analysis thereof, are keys for business success, and that they can be used to increase profits by leveraging the data to improve the existing applications and to create new applications (Couchbase, 2013). However, because of the increasing volume of data, and due to their constantly changing variety and to the use of cases thereof, the relational data model approach adopted by the traditional databases was pushed beyond its limits (Indrawan-Santiago, 2012).

Businesses were faced with the challenge of managing a large volume of unstructured or semi-structured data, with RDBMS traditionally not supporting this type of scalability (Mohan, 2013). NoSQL databases have an advantage, if compared to relational databases, due to the ability of NoSQL databases to handle unstructured data efficiently (Xiang *et al.*, 2010; Okman *et al.*, 2011). The relational database was also not designed to support changing data structures, and did not provide the flexibility that was required by these businesses (Adrian, 2012). Also, certain applications require very quick response times, and traditional RDBMS was unsatisfactory in providing the required performance (Mohan, 2013).

The limitations of the relational database as regards providing the required flexibility and scalability to handle big data have driven the process to acquire new innovative solutions for data processing (Adrian, 2012).

2.2.3 One size does not fit all

Stonebraker, Madden, Abadi, Harizopoulos, Hachem and Helland (2007) conclude in their paper ‘The end of an architectural era (it’s time for a complete rewrite)’ that the RDBMS attempts to be a ‘one-size-fits-all’ solution, and by doing so, it does not excel at anything. Stonebraker *et al.* (2007) argue that the RDBMS should be

redesigned, because it was architected more than 25 years ago. The RDBMS did not undergo an architectural redesign corresponding with the huge changes that were observed within the hardware sector. The speed and the capacity of processors, memory, and hard disks have increased enormously over the past 25 years, but the characteristics of a RDBMS can still be traced to System R, which is a database system that was designed in the 1970s.

The data requirements of different applications are also not identical, according to North (2010). The needs for “data, query and index types” that are required by such applications as “online transaction processing, business intelligence, customer relationship management, document processing and social networking” are not the same. The same can also be said for their “consistency, scalability, flexibility and security” requirements (North, 2010). No perfect solution exists that can meet the needs of all the different varieties of applications and their data requirements (Cattell, 2011).

Due to the data requirements of applications not being the same, a large number of NoSQL databases have been developed in order to solve the different data requirements. The different NoSQL databases do not attempt to be a ‘one-size-fits-all’ solution but are rather specialised to be used for certain data problems (Hecht & Jablonski, 2011). Therefore, the data requirements of the applications will determine which NoSQL database will be best suit the specific circumstances.

The data that are becoming available in the world, as explained in the previous section, are growing to massive levels. In terms of the increasing volume of data and the need to process the data within an increasingly shorter period of time, the traditional RDBMS was found to be lacking. Shalom (2009a) states that, for these reasons, people were forced to think of an “alternative approach to the traditional database”. With the realisation that different data problems could not be efficiently solved by using the same solution, the idea that the relational database could be a ‘one-size-fits-all’ solution did not stand firm. Shalom (2009a) concludes that the thought that ‘one size fits all’ was, and still is, wrong.

2.2.4 Cloud computing

The norm according to which business and consumer applications operated has recently changed from single-user applications to applications supporting many users. The Couchbase (2013) White Paper explains this shift as a move away from two-tier, client-server architecture, to three-tier Internet architecture, consisting of a web browser, or mobile application, tier, a web or application server tier, and a database server tier. The relational database was initially the first choice for the database tier, but did not fit very well within the highly distributed nature of the three-tier Internet architecture. Figure 2.1 below depicts this shift.

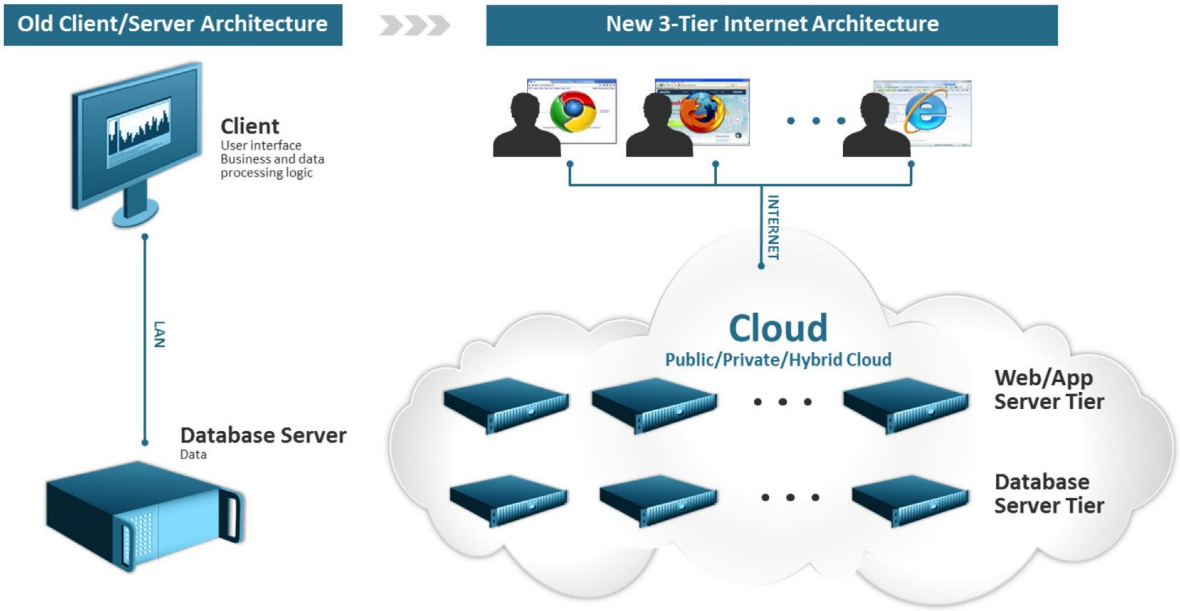


Figure 2.1: Illustrative description of the shift from a two-tier, client-server architecture to a three-tier Internet architecture (Couchbase, 2013: 3)

Cloud computing increased the desire by businesses to make use of commodity servers that fit better within the distributed nature of the three-tier Internet architecture. Commodity servers tend to be less expensive and easier to manage than is the implementing of a larger server with better hardware specifications, as is often the requirement when scaling a RDBMS. Such parameters required that

databases should support easier scalability than the traditional relational database did (Couchbase, 2013; Mohan, 2013).

According to Pokorny (2013), cloud computing requires a database with a flexible data model that is able to store structured and unstructured data. Pokorny further describes the following problems for cloud computing providers, which are related to their database requirements:

- data consistency;
- availability;
- predictable performance;
- scalable and high performance storage.

Because the traditional relational database did not provide a data model with the required flexibility, and were unable to effectively solve the database requirement problems of cloud computing providers, there was an increasing need for alternative database solutions that could.

Also, with the emergence of cloud computing, the limitations, as experienced by the large Web 2.0 companies, became much more common. This led to a demand for alternative database solutions that prioritise scaling and that are more cost-effective (Shalom, 2009b).

2.3 Characteristics of NoSQL

The drivers, as discussed in the previous section, forced some businesses to seek alternative data solutions. NoSQL databases were designed to solve the data problems for which a relational database was deemed to be inadequate. The extensive variety of NoSQL databases in existence were not all designed to solve the same problems, in keeping with the idea that ‘one size does not fit all’. Therefore, some NoSQL databases do not have much in common, apart from their not using a relational model (Evans, 2009; Leavitt, 2010).

However, most NoSQL databases can be described as having similar characteristics. Their main characteristics can be defined as follows:

- not adhering to the relational model;
- designed to scale horizontally;
- not providing full ACID (atomicity, consistency, isolation, durability) guarantees; and
- possessing a dynamic data model (Indrawan-Santiago, 2012).

Not all NoSQL databases support the above-mentioned characteristics to the same extent, due to the wide variety of NoSQL databases that have been developed to solve the various data problems. Nevertheless, it can be argued that these characteristics are widely accepted as the NoSQL commonalities.

The characteristics identified, and also how they differ from a relational database, will be discussed in greater detail in the following subsection.

2.3.1 Non-relational model

In 1970, Codd outlined a relational approach for databases in his paper ‘A relational model of data for large shared data banks’, with the model having since become the dominant form in the database market (Indrawan-Santiago, 2012). In contrast, the NoSQL database community is united by the use of a non-relational model (Leavitt, 2010).

The way in which data are modelled within a relational database did not meet the needs of the large Web 2.0 companies (Mohan, 2013). Data within a relational database are usually normalised, which involves efficiently organising the data within a database. To satisfy end-user requests, the relational model generally requires joint operations that can be very resource-intensive, according to the Datastax (2013a) White Paper. In contrast, data within NoSQL databases are not normalised, but, rather, they are organised in a variety of different structures (Datastax, 2013a).

The process of normalisation, in which storage space is reduced by efficiently organising data, was first introduced when storage was seen as a limiting factor. The dramatic decrease of hardware prices, specifically the price per gigabyte, has helped to ensure that storage is cheap and abundant. Normalisation, though, can come at the expense of data retrieval complexity, because normalisation often requires data to be separated into many different tables. Doing so increases the complexity of data retrieval, as the data require retrieving from the many interrelated tables (Couchbase, 2013).

Also, as a direct consequence from NoSQL databases not adopting the relational approach, SQL cannot be used as a query interface. SQL was developed to be used with relational databases, therefore non-relational databases are unable to handle the SQL query interface (Bartholomew, 2010). There are no standardised query language for NoSQL databases and the query functionality and capabilities provided by the variety of NoSQL databases differ significantly (Hecht & Jablonski, 2011).

The most common categories of data structures or data models used by NoSQL databases, and examples of systems that use them, according to the http://nosql-database.org website, are:

- the key value pair – DynamoDB, Riak, Voldemort;
- the column family – Cassandra, Hypertable, Amazon SimpleDB;
- the document store – MongoDB, CouchDB, RavenDB; and
- the graph database – Neo4J, Infinite Graph, HyperGraphDB

This research assignment will not provide specific details of the different types of data structures or details regarding the variety of NoSQL solutions as it falls outside the scope of the research.

2.3.2 Scale horizontally

Scaling, in database terms, refers to the ability of a database to handle an increasing amount of data and/or users. This is usually achieved by scaling vertically (scale-up), or horizontally (scale-out).

A relational database traditionally scales vertically, by adding bigger or more processors, memory, and disk storage to the server running the database. The process of scaling vertically tends to be disproportionally expensive when designing, building, and supporting the database. The process of scaling vertically increases the operational complexity significantly, and the related licensing costs of a commercial relational database can be a significant IT expenditure for a business. A relational database running on more than one server usually uses replication to keep the database synchronised, and requires sharding to scale, which involves splitting up the databases into different tables, which is considered to be an inherently complex process (Lai, 2009; Padhy, Patra & Satapathy, 2011; Couchbase, 2013; Nance, Losser, Iype, & Harmon, 2013; Pokorny, 2013).

In contrast, NoSQL databases are designed to scale horizontally, and to be processed in a distributed manner (Indrawan-Santiago, 2012). A cluster of servers that can be standard, physical, or virtual, can be used by NoSQL databases to store the available data. The database scales by adding additional commodity servers to the cluster, allowing the data to be horizontally partitioned. Commodity servers are inexpensive when compared to large commercial servers. The expensive licensing costs of commercial relational databases can also be avoided when a business opts to use open-source NoSQL alternatives (North, 2010; Cattell, 2011; Bonnet, Laurent, Sala, Laurent & Sicard, 2011; Couchbase, 2013; Pokorny, 2013).

As NoSQL databases are designed from the start to scale horizontally, the database intrinsically provides easier scalability capabilities than does their relational counterpart. Even though the scaling strategies of the vast array of NoSQL databases differ, automatic sharding capabilities are provided by the more advanced NoSQL databases, and therefore the complexity that comes with it, is avoided (Lai, 2009; Floratou *et al.*, 2012; Datastax, 2013a).

2.3.3 No ACID guarantees

ACID is a set of properties that is used by databases to guarantee the integrity and reliability of data. Such a system was implemented in relational databases by means of the use of a locking mechanism. The requirements of ACID pose strict rules on how transactions are processed, with the rules being entrenched deep into almost every relational database (Indrawan-Santiago, 2012; Bosworth, 2013). Adrian (2012) describes each of the components of ACID as follows:

- Atomicity – The database transaction is treated as one unit, therefore if all the update actions did not succeed, the entire transaction will be rolled back.
- Consistency – Every piece of data of the database transaction should be valid, which means that the data of the transaction should not break any of the defined database rules, and after the transaction, the database will still be in a consistent state.
- Isolation – Database transactions are kept separate and do not interfere with each other, which means that transactions are executed concurrently.
- Durability – The database transactions do not change after they have been executed and the transactions will therefore survive failures of servers or storage.

NoSQL databases do not provide full ACID guarantees. The strict rules that pertain to applying ACID properties to databases are considered unnecessary and too stringent in some distributed databases (Lai, 2009; Indrawan-Santiago, 2012). Relaxing ACID requirements can also improve the scalability and performance of the database (Cattell, 2011; Adrian, 2012).

The CAP (consistency, availability, and partition tolerance) theorem, based on Eric Brewer's (2000) ideas, and theorised by Gilbert and Lynch (2002), showed that ACID properties could not be guaranteed within distributed systems (Indrawan-Santiago, 2012). The CAP theorem states that one cannot achieve all three of the identified properties within a distributed system. Therefore, there are three configurations possible based on the CAP theorem (Bonnet *et al.*, 2011):

- consistency and partition tolerance (CP);
- partition tolerance and availability (PA); and
- consistency and availability (CA)

Therefore, in order for NoSQL databases to achieve availability in a partitioned environment, the consistency constraint is removed. However, some NoSQL databases do trade availability for consistency, for example HyperTable (Bonnet *et al.*, 2011). When consistency and availability are required, the database should not be partitioned (Indrawan-Santiago, 2012). Figure 2.2 below depict the trade-offs with regards consistency, availability and partition tolerance according to the CAP theorem:

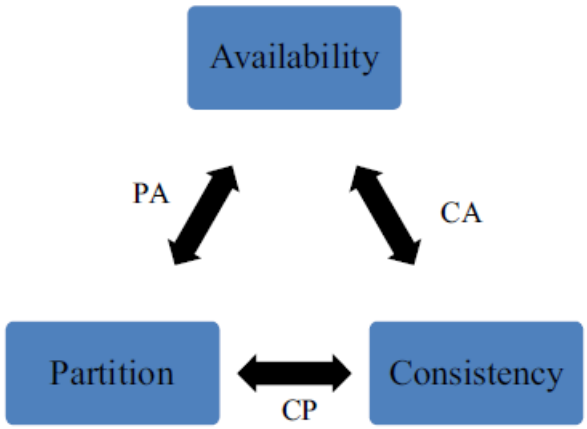


Figure 2.2: Illustrative description of Brewer's CAP theorem (Indrawan-Santiago, 2011: 47)

The consistency that is described by ACID properties and by the CAP theorem have different meanings, though. In contrast to the consistency requirement, as defined within the ACID properties context, the CAP theorem describes consistency within a distributed system. According to Adrian (2012), "[i]t describes a situation where different requests arrive at multiple locations in a system where the links between the locations has failed. The services will respond to preserve availability, but one of the

responses will be inconsistent; it is not possible to retain both complete consistency and complete availability throughout those failures. If the system is able to repair inconsistencies, the term 'eventual consistency' is used to describe its capabilities."

Due to Web 2.0, big data and cloud computing, as described above, partition tolerance have become inevitable and therefore the only viable comprise is that of the availability and consistency properties with regards the CAP theorem (Okman *et al.*, 2011). This has resulted in distributed database systems being designed, with reduced requirements of consistency, that support BASE (basically available, soft-state, eventually consistent) properties, rather than the traditional ACID properties (North, 2010; Padhy *et al.*, 2011).

2.3.4 Dynamic data model

Wikipedia describes a database schema of a database system as "the structure described in a formal language supported by the DBMS [that] refers to the organisation of data as a blueprint of how a database is constructed" (Wikipedia, 2013a). In a relational database, the schema has to be defined before data can be stored, and changing the schema thereafter, or evolving the schema further, is a complex and difficult task to perform (Couchbase, 2013).

This rigid model was abandoned by NoSQL developers, as it became increasingly difficult for databases to accommodate for new types, as well as for new use cases for the data. Web 2.0 businesses especially required data models that could evolve with their changing requirements without requiring any downtime, and the relational model, even with recent enhancements, did not provide the necessary flexibility (Bartholomew, 2010; Bonnet *et al.*, 2011; Hecht & Jablonski, 2011; Mohan, 2013).

When new types of data become available through the different sources that had been developed with the emergence of the 'big data era', a flexible data model was essential in order to integrate and leverage the new data seamlessly. Modern NoSQL databases provide a data model that can be adjusted without disrupting the normal operation and availability thereof (Cattell, 2011; Couchbase, 2013; Datastax, 2013a).

The relational database was also not a good fit for storing data structures that were similar to object-orientated programming languages, and therefore object-relational mapping was developed as a tool in order to make this possible. Such mapping requires that the object-orientated data requiring to be stored be manipulated in such a way that they can fit within the relational parameters. According to Hecht and Jablonski (2011), trying to store data, which does not match the relational mode, in a relational database, results in the use of “expensive mapping frameworks and complex algorithms”. NoSQL provides a much better solution for storing object-orientated code with their flexible data model than was previously available, and therefore also avoids the time-consuming process of object-relational mapping (Lai, 2009; Strauch, 2011).

2.4 Benefits of NoSQL

The drivers and characteristics of NoSQL have been discussed up to this point so as to provide an understanding of the alternative NoSQL databases. The following part discusses some of the key benefits that can be achieved over the relational database when opting to follow the NoSQL route.

2.4.1 Increased performance

The performance of any system, including a DBMS, has always been an important issue to consider prior to implementation. NoSQL database developers tend to market their database systems as providing increased performance over the traditional relational database.

One of the goals that was set and achieved by Google for Bigtable was for it to be a high performance distributed storage system (Chang *et al*., 2008). Cassandra, a data store that was developed by Facebook to implement a new search feature, has achieved write speeds that are 2500 times faster than MySQL (Lakshman & Malik, 2009). Amazon’s Dynamo is designed to make trade-offs between availability, consistency, cost-effectiveness, and performance (DeCandia *et al.*, 2007). According to Catell (2011), NoSQL databases efficiently use distributed indexes and computer

memory in order to improve performance. The performance gain that can be achieved when using NoSQL databases makes it worthwhile for businesses to consider them as a database solution.

However, according to a study performed by Floratou *et al.* (2012), an optimised RDBMS is still capable of outperforming NoSQL databases in some cases (Nance *et al.,* 2013). Therefore, even though some NoSQL database implementations have proved to be capable of delivering very high performance, an optimised RDBMS still has some advantages.

2.4.2 Reduced costs

As was previously mentioned, many NoSQL databases are open-source. Therefore, the licensing costs that are associated with commercial relational databases can be avoided when moving to a NoSQL open-source alternative.

The commodity servers used by NoSQL databases in order to scale horizontally are also less expensive in comparison to the large commercial servers of the traditional relational databases, which are warranted when the scale of the relational database becomes very large (Cattell, 2011; Couchbase, 2013).

2.4.3 Reduced complexity

The complexities related to the sharding of databases are avoided by the more advanced NoSQL databases, because these databases provide automatic sharding capabilities (Floratou *et al.*, 2012). The horizontal scaling approach adopted by NoSQL database alternatives further reduces the complexity with regards scaling, as commodity servers can be added or removed with relative ease in order to accommodate any scaling requirements of the database (Lai, 2009; Cattell, 2011; Pokorny, 2013).

The data models used by NoSQL databases can also much more easily accommodate any new type of data, including object-orientated data (Hecht & Jablonski, 2011). Therefore, they require neither changing of the schema, nor the

implementation of object-relational mapping, as is the case with a relational database (Couchbase, 2013). Therefore, because NoSQL databases do not implement the strict rules of the relational schema, the flexibility of the data model increases. The flexibility of the data model results in the reduced complexity with regards operating the database when the requirements of the business change (Bartholomew, 2010).

2.5 Conclusion

In this chapter the drivers, the main characteristics, and the benefits of NoSQL databases were discussed. Also, during the discussion, the key differences of NoSQL and relational databases were indicated.

It is clear from the information above that the drivers of NoSQL had significant influences on the characteristics of NoSQL databases and the benefits of NoSQL databases are a direct consequence resulting from the needs that developed from the drivers of NoSQL. This discussion only focused on the key drivers, characteristics and benefits applicable to NoSQL databases in general. Therefore, the specific drivers, characteristics and benefits of the different types of data structures available within NoSQL databases, as well as specific NoSQL database solutions, where not discussed, as this falls outside the scope of this research assignment. The following papers provide more detail regarding different NoSQL database solutions and the data structures that they implement: Strauch (2011), Padhy *et al.* (2011), Hecht and Jablonski (2011), the MongoDB (2013) White Paper and the Datastax (2013a) White Paper.

Based on the discussion from this chapter, it becomes clear that NoSQL databases are more suited for Internet-based businesses. The characteristics of NoSQL databases can be more beneficial to, and provide a better fit for, the requirements of certain Internet-based businesses than can relational databases. However, Brooks (2011) states that NoSQL databases can be considered for adoption by businesses of all sizes. NoSQL databases might not be able to replace every business's relational database, but a business can also benefit from using a NoSQL database

as a hybrid solution with a relational database. NoSQL databases can therefore also be used as an additional tool in order to meet business goals.

3. NoSQL business imperatives

3.1 Introduction

Any business attempting to be successful should set up strategic goals and objectives that are unique to it. The business should pursue these goals and objectives diligently, as it is imperative that they are met in order for the business to succeed.

The strategic goals and objectives set by a business should be aligned with the IT goals of the business, in order to achieve effective IT governance (IODSA, 2009) and to reduce the gap between IT and the business. Such alignment can be achieved by identifying the business imperatives involved. Said imperatives can be defined as “those thrusts of activity that are critical to arriving at the stated objectives (complete and utter fundamentals). These are the drivers of the business, not themes of activity. They are the principles by which the business acts and thinks, not the actions it takes. There will be no dynamic display of information in this segment but these statements represent the high level forces that will move the enterprise to the desired future vision from their current reality and achieve the strategic outcomes and drive the redefined conditions where they will be absolute key performance indicators” (Group Partners, 2008; Boshoff, 2012).

For the purposes of this study, the business imperatives that are applicable to a business that is considering the implementation of a NoSQL DBMS are identified, based on the literature review, in an effort to reduce the IT gap and to provide IT and business alignment, as follows:

- zero downtime;
- hyper performance;
- massive scalability; and
- dynamic flexibility.

The business imperatives involved are discussed in the following section of this research assignment.

3.2 Zero downtime

The reliability of a business's IT systems, including its database, is always a matter of high priority, given the fact that most businesses are dependent on IT for the continuation of normal operations. However, some businesses tend to rely more on their systems than do others, as is the case with Web 2.0 companies. For a Web 2.0 company to have zero downtime is advantageous to their business. As such, it was one of the most important goals set by Amazon and Google when they developed Dynamo and Bigtable (DeCandia *et al.*, 2007; Chang *et al*., 2008). As Web 2.0 companies cannot deliver their services in the absence of the availability of their database, downtime can have significant financial implications, and can impact on customer trust.

NoSQL databases are designed to provide a highly reliable database environment, with zero amount of downtime. The ability of NoSQL databases systems to survive the failure of any of their components (software, server and network) ensures their high availability (North, 2010). The effective use of commodity hardware for distributed storage can provide more reliability than can high-end relational databases (Shalom, 2009b).

High availability is usually achieved by designing the database configuration to have no single point of failure. The Datastax (2013a) White Paper points out how this can be achieved by a NoSQL database, as can be seen below:

- All the nodes in the database cluster should be able to serve in the "same capacity (i.e. no 'master' node)".
- The NoSQL database is able to replicate and to segregate data easily between different physical racks in a data centre (to avoid hardware outages).
- The NoSQL database is able to support data distribution designs. The distribution can either be multi-data centre or on-premise, and in the cloud.

Bigtable and Dynamo have succeeded in providing Google and Amazon with reliable database solutions. Dynamo was designed to deal with the failure of components within the infrastructure as their “standard mode of operation” (DeCandia *et al.*, 2007), therefore providing the ‘always online’ experience required by Amazon’s business and customers.

Netflix, an online streaming business, made the decision to move to ‘high-availability storage systems’ when the risks of them continuing to use a single data centre were deemed to be too high. The avoidance of any outages and the negative impacts that they would have on customers was deemed imperative (Anand, 2010). Netflix decided to use Amazon Web Services’ SimpleDB and S3, an infrastructure-as-a-service solution provided by Amazon, to ensure the availability required by the business and their customers.

As was discussed in Chapter 2, the CAP theorem states that a trade-off exist for consistency and availability for a partitioned database. Therefore, in order to improve the availability of a partitioned database, the consistency requirements of the database should be reduced. Database systems that reduce the consistency requirements to achieve higher availability implements a BASE model rather than the traditional ACID properties. Social media websites and internet blogs can be eventually consistent, due to availability being the most important feature. This is in contrast to financial systems that require full consistency at all times (Hecht & Jablonski, 2011).

Bosworth (2013), the chief executive officer (CEO) of Datastax, explains the misconception that NoSQL databases can only be used by a “niche subset of applications”, and that the rest should use ACID-compliant databases. Developers of online applications require a database that is reliable and available due to the users, devices and sensors on which their applications rely always being online. Adrian (2012) and Bosworth (2013) also point out that the consistency of ACID and BASE do not mean the same thing, and therefore the two definitions should not be confused.

Based on the successful implementation by numerous businesses of NoSQL databases to gain increased reliability, availability and near zero downtime, NoSQL can be an appealing option. However, the first thing for a business to determine

should be whether zero downtime is considered to be a business imperative. This could be determined by measuring to what extent any downtime will affect the business, and whether the slightest moment of downtime will have significantly unfavourable business implications.

3.3 Hyper performance

The rate of data generation has increased significantly due to the 'big data era'. The impatience of customers, especially customers of web business, has also pressurised businesses to deliver their services without delay. During a test performed by Amazon, which caused a 100 millisecond delay for the users of their website, sales dropped by 1%. Google also experienced a drop of 25% in traffic when the search results, presented by the Google search engine, were increased to 30 items instead of 10, resulting in a corresponding delay of 900 instead of 400 milliseconds (Linden, 2006; Mayer, 2009 (as cited by Hecht & Jablonski, 2011)). Therefore, when a business is faced with huge amounts of data that require quick storage and accessing, as well as with impatient customers, the performance of the database to meet users' needs becomes an imperative.

By not providing full ACID compliance, NoSQL databases are able to provide a higher level of performance, compared to the ACID complaint databases (Adrian, 2012). Bigtable and Dynamo, and the many NoSQL databases developed thereafter, were designed to be high-performing database solutions. Cassandra, a NoSQL database solution, offers the ability to increase performance in an almost linear fashion, by means of adding commodity servers to the database cluster (Datastax, 2013a).

The data structures used by NoSQL databases are also a better fit for object-orientated code, therefore delivering higher performance than does a relational database. The reason for this is that it is not necessary to manipulate the data before storage can take place (Lai, 2009).

Therefore, when full ACID guarantees on your database transactions are considered less beneficial than is the performance of the database, a NoSQL database can provide a better solution to meet the needs of the business.

3.4 Massive scalability

Scaling, in order the meet growing data and user needs, has become a much more common issue through the emergence of Web 2.0 businesses, big data, and cloud computing. Businesses operating within these domains are not only required to scale to massive levels, but also to do so efficiently. The ability to accommodate large users and to manage fast-growing data sets places high importance on the scalability requirements of databases. The negative implications of being unable to scale to the levels required by the business can include downtime and reduced performance.

NoSQL database developers design their databases, from the start, with the ability to scale horizontally and replicate and partition data over many commodity servers (Cattell, 2011; Indrawan-Santiago, 2012; Floratou *et al.*, 2012). The design choice of scaling horizontally enables NoSQL databases to scale to massive levels much more easily than it was possible to do with relational databases. When they are required to scale, commodity servers can easily be added (or removed) to meet the requirements of a business (Floratou *et al.*, 2012). According to the Datastax (2013a) White Paper, an “enterprise-class NoSQL solution” is capable of handling multiple data centres and of determining where “read and write operations occur” automatically. Therefore, the complexities of operating a distributed database system are greatly reduced.

NoSQL databases therefore are able to meet the unpredictable demands for computing resources experienced by businesses by providing an easy solution for scaling (North, 2010).

3.5 Dynamic flexibility

The huge variety of data sources that is available delivers an increasing variety of structured and semi-structured data. The possibilities available within the data can provide businesses with a competitive advantage by enabling them to use said data in innovative ways. Therefore, the challenge for a business is having the flexibility that is required to leverage new types of data, by incorporating them within their existing database, so that they can arrive at innovative new ways of using them to

provide business profits, and to meet the goals of the business. The data model of the database should provide the flexibility that is required to enable the incorporation of new data types, without having to disrupt the normal operation of the database that might result in downtime.

NoSQL databases implement data models that support flexible schemas, thereby enabling changes to take place to the schema without affecting the availability of the database (Cattell, 2011; Datastax, 2013a). The flexible data models of NoSQL databases can therefore provide a better fit than fixed schemas used by relational databases. This flexibility is required in order to effectively handle the agile nature of the Web 2.0 web applications (Hecht & Jablonski, 2011). The flexibility of the data models implemented by NoSQL databases also provides them with the capability to store unstructured data, and this is seen as one of the main advantages that NoSQL have over relational databases (Xiang *et al.*, 2010; Okman *et al.*, 2011).

With the flexible data models of NoSQL databases, new data types can easily be incorporated, enabling users to leverage them effectively.

3.6 Conclusion

In this chapter, the business imperatives that are applicable to a business that can benefit from implementing a NoSQL database solution were discussed. The identified business imperatives can assist businesses to determine whether a NoSQL database is a viable option, and can also provide guidance on the aligning of the business and IT objectives. The first step in aligning business and IT objectives is to identify the imperatives of the specific business. Thereafter, if the imperatives are similar to the business imperatives that were discussed in the course of this chapter, the business would be able to benefit from implementing a NoSQL database, and the viability of implementation should be considered. The business imperatives as discussed within this chapter can serve as the starting point for a business to align their strategic business objectives with the goals of IT.

However, the enumeration of business imperatives contained in this chapter should by no means be seen as the only imperatives that would be applicable to NoSQL databases. Only the most significant incremental business imperatives related to the use of NoSQL databases, as identified from the literature review, falls within the

scope of this research. Therefore, generic business imperatives related to databases in general were not discussed.

4. NoSQL strategic IT risks

4.1 Introduction

In Chapter 2 of the current research assignment, key drivers, characteristics and benefits were discussed before the business imperatives that are related to NoSQL databases for business were identified. In this chapter, the risks that are related to implementing a NoSQL database in a business will be discussed from a strategic IT perspective.

The strategic IT goals of a business should support the strategic business goals set by the business. In order to achieve the strategic IT goals, the business is required to perform strategic IT planning that supports its goals (COBIT 5, 2012). Strategic risks, for the purposes of this study, can be defined as the risks that a business faces that will impact on its strategic IT plan, and which might result in the strategic IT goals not being achieved.

In this chapter, the strategic IT risks related to NoSQL databases will be discussed in terms of the following types of risk:

- acquisition risk;
- vendor sustainability risk;
- retrofitting risk;
- alignment risk;
- skills risk; and
- interoperability risk.

The risks discussed in the course of this chapter should not be seen as a complete and comprehensive list of strategic risks that might impact on a business. However, the strategic risks were identified as the most important risks. Therefore, the extent to which they might impact on a business should be determined.

4.2 Acquisition risk

The acquisition risk refers to the risk that the management of the business might make the incorrect choice regarding which NoSQL database to implement. This risk will always be present, and, in most cases, a business will only be able to be completely sure whether the correct decision has been made by practising hindsight. However, the management should ensure that they have a thorough understanding of the implications of the acquisition risk involved.

The lack of standardisation of NoSQL databases can pose a significant acquisition risk for a business. According to Oliver (2012), relational databases provide, at the very least, an ANSI (American National Standards Institute) standard SQL and an Open Database Connectivity (ODBC) connector standard. NoSQL databases, in contrast, lack standardisation. He further states that the lack of standardisation within the NoSQL database market increases the difficulty of changing from one NoSQL vendor to another (Oliver, 2012). It is therefore important that a business implements a NoSQL database that is able to meet the business requirements, to prevent the issues that may arise from changing to another database at a later stage.

Also, the different data structures (key value pair, column family, document store, and graph database) used by NoSQL database vendors require consideration by the management of the business. The different data structures provide solutions for different data problems, and they have different strengths and weaknesses. The situation is further complicated due to the data requirements of a business, which can change over time. The change in the data requirements of a business can result in another data structure providing a better fit for meeting the needs of the business (Oliver, 2012). Due to the difficulty encountered in changing NoSQL vendors, a business might not be able to change to another NoSQL vendor with a data structure that better fits the requirements of the business.

The huge variety of NoSQL database solutions, with each having its own strategies for providing performance, scalability, availability, and flexibility, further complicates and increases the acquisition risk. The unfamiliarity of NoSQL databases by decision-makers of business tends to make them side step NoSQL completely. This is because the decision-makers do not have sufficient knowledge of the subject in order to make an informed decision (Nance *et al.,* 2013).

Without understanding the implications of the acquisition risk for the business, the management involved cannot make informed decisions to mitigate the risk to an acceptable level. Because of the huge variety of NoSQL databases that have been developed to solve the different data problems, and no solution exists that meets all the requirements, it is a difficult task to select a system that will best suit the specific circumstances of the business (Hecht & Jablonski, 2011). Therefore, management should ensure that the required researched is performed, in order to gain sufficient knowledge, which will enable them to make the best possible decision, regarding which NoSQL database to implement.

4.3 Vendor sustainability risk

Due to the relatively new concept of non-relational databases, and therefore also NoSQL databases, most NoSQL developers are still in the start-up phase of their business. The NoSQL vendors have not yet reached the same level of maturity as have the commercial RDBMS's vendors (Adrian, 2012). This therefore increases the risk of vendor sustainability and the many negative impacts that it can have on a business.

The many negative impacts that the sustainability of vendor can have on a business can range from no support being available for the database, to the NoSQL database becoming obsolete. According to the MongoDB (2013) White Paper, strong commercial support and community strength are key indicators with regards the sustainability of the vendor. The benefit for a database, due to the strong commercial support it receives, is not only that the sustainability risk will decrease, but it can also positively impact the evolution and features thereof. The strength of the community of the database will impact the availability of developers that are familiar with the product, and how easy database information, documentation and code samples can be obtained (MongoDB, 2013).

The management of a business should therefore consider the risk of the NoSQL vendor's sustainability to mitigate this risk.

4.4 Retrofitting risk

Retrofitting refers to the addition of new features to a system. This will usually be performed to enable the system to meet the specific needs of a business. The retrofitting risk refers to when these additions or modifications to the system are performed to such an extent that, in the worst case scenario, the system becomes obsolete. Some of the reasons for the retrofitting risk occurring, as applicable to NoSQL databases, are discussed below.

Many NoSQL database solutions are open-source; therefore users are able to make modifications to the source code of the database in order to suit the needs of the specific business (Mohan, 2013). However, having access to the source code and modifying it can increase the likelihood that a retrofitting risk will occur.

Modifying the source code of a database can either increase the complexity of installing software updates released by the developer, or make the database unable to update at all. This is because the source code does not function in the same way as does the original database. The extent of the modifications to the system determines how difficult or time-consuming it is likely to be to update. The update process will in all likelihood overwrite all source code modifications, requiring them to be re-performed. If the system were to be modified to such an extent that re-performance of all source code is not a viable option, the system will be unable to update. Updating software is very important, as security and other issues are usually fixed by such a means. The benefit of having an open-source product, to which the open-source community can contribute in order to improve the product (Datastax, 2013a), is also greatly reduced when these improvements cannot be implemented by updating the system. Therefore the inability to update the database can pose significant risks for a business, and might also result in the system becoming obsolete.

Even though modifying the source code of the NoSQL database to better fit the requirements of a business can prove to be beneficial for the business, care should be taken when doing so, and the risks thereof should be understood.

4.5 Skills risk

The skills risks refer to the risk that the skills that are required to implement, to operate, and to maintain the database to meet the business requirements is unavailable. Due to the NoSQL database still being relatively new within the database market, and to it not yet having been widely adopted (Adrian, 2012), the expertise and the experience within the field is still very limited. Developers are still struggling with all the specific characteristics of NoSQL databases (Hecht & Jablonski, 2011). Therefore, businesses might find it difficult to acquire the necessary skills, which can pose a significant risk for a business.

The functions and the features of NoSQL database solutions are not as advanced as are those of the established commercial relational databases. NoSQL databases generally require specialist programmers with the necessary skills to create the required business logic. Adrian (2012) explains that “this refactoring of logic requires different skills, and can be more error-prone and fragile, and harder to maintain, than the centralized control relational DBMSs (RDBMSs) provide”. Implementing additional query functionality to a NoSQL database is often required to be performed on the application layer. However, adding more query functionalities to a NoSQL database can quickly introduce many complexities to the system, as well as impact performance (Hecht & Jablonski, 2011).

Due to the fact that most NoSQL databases do not support high-level query languages, data optimisation can only be performed by those with the necessary programming skills. Also, the many sophisticated optimisation technologies that have been developed over the years, and which have been built into commercial relational databases, should now be performed by programmers with specialist skills (Mohan, 2013).

According to Indrawan-Santiago (2012) and Anand (2010), there is a general lack of knowledge regarding NoSQL database solutions. Both the benefits and the limitations of NoSQL databases are not fully understood by the decision-makers of businesses. Indrawan-Santiago (2012) also mentions this as a reason for NoSQL databases not yet having received widespread adoption.

Due to the open-source nature of NoSQL databases, not all NoSQL vendors offer the formal support that might be required by a business (Lai, 2009). A business should therefore ensure that its IT employees have the required skills in order to be able to support the database sufficiently. However, software companies have begun to provide commercial support for some NoSQL solutions (Indrawan-Santiago, 2012), and therefore have started to provide assurance as regards the support of NoSQL database solutions.

Such specialist skills and expertise might prove difficult for a business to acquire and retain, especially due to the market still being relatively new. Therefore, a business wanting to implement a NoSQL database solution should either ensure that the relevant expertise and skills, or a commercial software company that supports the NoSQL solution, is available to mitigate these risks. As discussed above, the strength of community of the specific database can be a key indicator with regards the availability of developers that are familiar with the database.

4.6 Interoperability risk

Interoperability is the ability of different systems to work together. Therefore, the interoperability risk occurs when the systems of a business are unable to communicate in the intended manner, or when they are only able to communicate after the interoperability issues have been solved.

When implementing a NoSQL database, businesses already operating a relational database are required to migrate from a relational database to the NoSQL database. Alternatively, some businesses can opt to use a NoSQL database as an additional tool, in conjunction with a relational database. However, as the migration process does not happen instantaneously, both scenarios are likely to require a business to operate both a relational and a NoSQL database. Such requirements might result in an interoperability risk, in terms of which the different database systems are unable to communicate with each other in the intended way, or as required by the business.

Mohan (2013) states that many businesses would be unable to perform such a major transformation as moving from an RDBMS to a NoSQL DBMS. Netflix was able to move from an RDBMS to a NoSQL DBMS. Netflix used a data replication system,

which they specifically developed to move their data from the relational database in use to the NoSQL databases of Amazon, SimpleDB, and S3. However, Netflix experienced many problems during the migration process, due to its complexity (Anand, 2010).

Also, NoSQL database solutions use an application programming interface (API) that the vendors develop specifically for their database (Mohan, 2013). However, the API protocol used might not be able to communicate with the other systems already in use by the business. Therefore, the developers are required to modify the API, or to develop an application to enable the systems to interoperate. Many businesses would not be able to perform such modifications, and would therefore require specialist skills.

Therefore, the management of a business should be aware of the interoperability risks, and should determine viable ways in which to solve the interoperability issues.

4.7 Alignment risk

The alignment risk refers to where the IT and strategic objectives of a business are not aligned. Therefore, the IT of the business, in such a case, does not completely support the business goals set by the latter. The business imperatives, as applicable to NoSQL databases and discussed in Chapter 3, form the basis for aligning the strategic and the IT objectives of a business. The strategic IT risks, as discussed up to this point during this chapter, can have a significant impact on the alignment of IT and strategic objectives. Therefore, the alignment risk is a direct result from the risks presented by the previously discussed strategic IT risks.

In the context of NoSQL databases, and when using the business imperatives identified in Chapter 3, the misalignment between the strategic and the IT objectives offer the following risks:

- the risk of downtime of the database, meaning that the business is unable to operate, resulting in financial loss;
- the risk that the business's database performance requirements are not met, with the result that the users' and the clients' expectations are not met;

- the risk that the database is unable to scale effectively and efficiently to the required level to accommodate the fast-growing numbers of users and clients; and
- the risk that the data model does not provide the required flexibility to allow for the business to incorporate new types of data. This might lead to lost opportunities, or to a product that is inferior to those of one's competitors.

The management should ensure that the IT and the business objectives are aligned in order to prevent the implications of the above-mentioned risks. The management should therefore assess the risks of the NoSQL database not meeting the requirements of the business imperatives, prior to the implementation thereof.

4.8 Conclusion

The strategic IT risks discussed in this chapter have a direct impact on the business imperatives identified during the previous chapter. The strategic IT alignment risk provides the link between the strategic IT risks and the business imperatives, and, also, the risk with regards misalignment of IT and strategic objectives.

Therefore, management should determine the impacts of the strategic IT risks on their business, in order that they can make suitably informed decisions with regards how to mitigate the negative impacts on the business imperatives. The strategic IT risks identified should be incorporated within the strategic IT plan of a business to ensure improved alignment of IT and strategic objectives. The risks should be thoroughly considered by the management involved to also determine the viability of using a NoSQL database as a solution for their business.

However, only the most significant incremental strategic IT risks, which will in most circumstances be present, as identified from the literature review performed, were discussed in this chapter. This is therefore not a complete and comprehensive list of all the possible risks that are available. The management of a business should also determine and consider any other strategic IT risks relating to the specific circumstances of their business.

5. NoSQL operational IT risks

5.1 Introduction

Every component within the IT environment is exposed to different risks, some of which are likely to be more important than other, and the operational IT employees of a business need to manage such components effectively. Therefore, a business is required to identify the operational risks that might have an impact on the business.

Operational risks can be classified within the following categories (Boshoff, 2013), which agree with the life cycle enabler dimension of COBIT 5 (COBIT 5, 2012):

- plan;
- design;
- set up/configure/build;
- operate; and
- maintain.

The operational risks that are related to NoSQL databases are discussed in this chapter. The planning and design risk categories are discussed generically. However, the other categories that are mentioned above are discussed within the context of Cassandra, which is an open-source NoSQL database. Due to the huge variety of NoSQL databases in existence, the risks within these categories can differ significantly, and therefore a NoSQL database was chosen to provide the context for the research.

5.2 Generic risks

5.2.1 Planning risks

Planning can be described as a “scheme, program or method worked out beforehand for the accomplishment of an objective” (The Free Dictionary, 2013). Within the context of IT, planning can include identifying objectives, determining the information architecture, developing standards, and defining definitions (COBIT 5, 2012). Determining the IT planning risks can provide key focus areas within the planning process.

The risks presented in Table 5.1 below were identified on the basis of the strategic risks, as discussed in the previous chapter.

Table 5.1: Operational IT planning risks

Strategic risk	Planning risks
Acquisition risk	Acquisition risk is the risk of insufficient planning that can result in the selecting of a NoSQL database that will not benefit the business the most. Inappropriate selection can occur due to the following factors: • The data model or data structure that best fits the requirements of the business was incorrectly determined. • The business requirements changed, and therefore the data model or data structure initially chosen was incorrect. • The NoSQL database does not provide the required performance, scalability, availability, and/or flexibility.

Strategic risk	Planning risks
Vendor sustainability risk	The vendor sustainability risk is the risk of insufficient planning that can result in the selection of a NoSQL vendor that becomes unsustainable. This can occur when the business makes an inappropriate assessment of the commercial support and/or community strength to determine the sustainability of the vendor (MongoDB, 2013).
Retrofitting risk	The retrofitting risk, as it relates to operational planning, is the risk that retrofitting, and the impacts thereof, were not properly considered and planned. This might be due to the responsible person(s) not having the required skills to perform the task, and/or their not understanding the business impacts thereof. Also, the features that were initially considered to be unnecessary could, if they are required at a later stage, be difficult to implement then (Mohan, 2013). This can give rise to the risk that the database can become obsolete.
Alignment risk	The alignment risk is the risk that the planning was not properly performed, so as to ensure alignment with the business goals. This might be due to the business requirements for availability, performance, scalability, and flexibility, as well as the way in which the requirements will be met, not being properly defined.
Skills risk	The skills risk is the risk that the business did not sufficiently plan to address how the necessary specialist skills would be acquired and retained, in order to operate the NoSQL database as intended.
Interoperability risk	The interoperability risk is the risk that the planning involved did not determine whether all the systems could interoperate, or how, and whether the business would be able to enable all the systems to interoperate, as required by the business.

The operational IT planning risks identified the links directly with the strategic IT risks, as discussed in the current chapter. Management should be aware of how the

operational IT planning risks can impact on the business, and also plan how they will respond to the risks.

5.2.2 Design risks

Design, in the context of NoSQL databases, includes the following aspects:

- Designing the NoSQL database from scratch: However, due to the variety of available NoSQL databases, in most cases doing so is not required, and should only be attempted by businesses with the required budget and resources.
- Designing, or retrofitting, the NoSQL database, by modifying the source code of the database to meet the needs of the business: This can only be performed with open-source NoSQL databases, and is likely to give rise to the retrofitting risk discussed in Chapter 4.
- Designing client applications to operate with the NoSQL database: Due to NoSQL databases supporting neither the same level of features, nor the high-level query languages with which relational databases come (Mohan, 2013), programmers should in many cases be required to design applications that can provide the required functionality.
- Data modelling design: This aspect of the design considers how the data will best fit into the data structure of the database.

The first item in the above list will not be discussed in this research assignment, because it is outside the scope of this research assignment. The second item was already discussed in Chapter 4, and therefore only items 3 and 4 will be discussed in this section.

Designing client applications for NoSQL databases requires people with the necessary skills who understand the business, and a programming language that is able to connect to the database. According to Hecht and Jablonski (2011) developers are still struggling with all the characteristics of NoSQL databases. Also, NoSQL databases do not have a strong theoretical background, as is the case with

relational database, as regards data modelling design (Indrawan-Santiago, 2012). Without the theoretical background, the data modelling design for NoSQL databases can be very challenging.

In light of the above, designing client applications and data modelling design requires specialist skills and expertise, and therefore the business is faced with skills risks, as were discussed in Chapter 4.

5.3 Cassandra-specific risks

Cassandra is an open-source NoSQL database solution that is classified under the column family. Cassandra is commercially supported by the software company, Datastax. As from 3 September 2013, it has been ranked as one of the 10 most popular database engines (Andlinger & Gelbmann, 2013). Cassandra boasts of the following characteristics (Datastax, 2013b):

- high scalability on commodity servers;
- the ability to handle structured, semi-structured, and unstructured data;
- high availability, with no single point of failure;
- high performance; and
- a highly flexible data model.

Cassandra employs a peer-to-peer distributed system, with data automatically being distributed to all the nodes, or computers, within the database cluster. Data replication capabilities come standard on Cassandra, and can be configured in order to achieve high availability, and no single point of failure. Performance can be enhanced in a linear fashion by adding more nodes to the database. Figure 5.1 below displays the ‘ring’ that Cassandra employs, and how it supplies linear scalability, therefore increasing the number of transactions per second (txns/sec) that the database is able to achieve in a linear fashion, by adding new nodes.

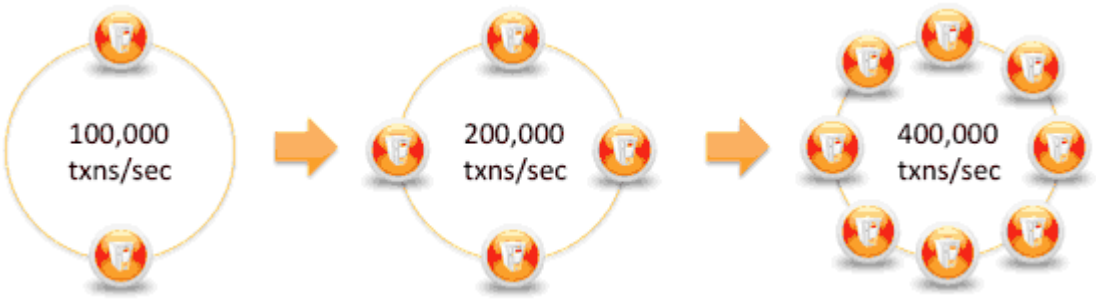


Figure 5.1: Cassandra's linear scalability (Datastax, 2013b)

The following sections will discuss the operational business risks related to Cassandra for setting up/configuration/building, operating, and maintenance.

5.3.1 Setting-up/Configuration/Building risks

From an IT perspective setting up, configuration and building can be defined as follows (Boshoff, 2012):

- Setting up – Setting up or installing a program onto a computer system, enabling it to be executed.
- Configuration – Creating configuration files that configure the initial settings for computer programs.
- Building (software) – Creating and converting source code files into stand-alone software artefacts able to run on a computer.
- Building (hardware) – Assembling various components, enabling it to accept an operating system and to function as a computer.

In the following table, the risks that are related to setting up, configuration and building in the context of using Cassandra are discussed. Also, the risk level that is associated with each action is determined. The detail, as listed under the "Cassandra perspective" column in the table below, was obtained from the Datastax website (Datastax, 2013b).

Table 5.2: Setting-up/Configuration/Building risks

Action	Cassandra perspective	Risk level
Setting up	This would entail installing Cassandra and all other required software that may include: • the Cassandra query language (CQL); • OpsCenter, the browser-based user interface; • Java driver, the driver that enables Java applications to connect to the database; and • C# driver, the driver that enables C# applications to connect to the database. As step-by-step instructions for how to install Cassandra are available on the Internet, the installation should be a relatively straightforward process.	Low
Configuration	The following list provides some of the main configurations that can be performed in Cassandra: • Initialisation properties control how a node is configured within a cluster, including internode communication, data partitioning, and replica placement. • Global row and key cache properties serve as cache parameters for tables. • Performance-tuning properties enable tuning performance and system resource utilisation, including memory, disk input/output (I/O), the central processing unit (CPU), reads, and writes. • Binary and remote procedure call (RPC) protocol timeout properties are timeout settings for the binary protocol. • RPC tuning properties consist of settings for configuring and tuning RPCs (client connections). • Fault detection properties are settings for handling	High

Action	Cassandra perspective	Risk level
	poorly performing or failing nodes. • Automatic backup properties provide automatic backup settings. • Security properties provide server and client security settings. Only someone with the required skills will be able to configure the database to operate to meet the business requirements. Due to the lack of standards and best practices regarding NoSQL databases, the configuration process can be complex to perform.	
Building (software)	Cassandra provides two drivers, Java and C#. Therefore, programmers can write applications to connect to the database using either of the two. In order for programmers to write database client applications, they should have expert knowledge of the programming language, as well as also understanding the business, to incorporate the business requirements within their applications. Cassandra does not provide the same extent features or high-level query capabilities that are available in commercial relational databases. Therefore, the programmers need to build client applications, as required by the business.	High
Building (hardware)	Cassandra can easily scale using commodity servers, which greatly reduces the complexity of the process, as standard computers can be used. Therefore, doing so does not pose a significant risk.	Low

Based on the above, the highest risk areas are those of configuration and software build. The reasons for the risks involved were determined to be:

- the lack of standards and best practices; and
- specialist skill requirements

The lack of standards is not only a Cassandra-specific issue, but applies to all NoSQL databases. Oliver (2012) states that the lack of standards is one of the reasons why businesses are reluctant to move to NoSQL databases. Also, according to Adrian (2012), programmers are required to "create data models and sophisticated business logic", because business users are unable to do so because they lack the required skills.

5.3.2 Operational risks

Within the context of databases, operating can include a broad number of actions. The business users normally operate with the database by performing such tasks as handling queries and reporting. IT users normally do not operate with the database, but, rather, tend to perform more maintenance-orientated tasks.

Therefore, the design of the database and the applications built by the programmers tend to determine how business users operate with the database. Operating is therefore dependent on the previous sections, and will not be further discussed in this research assignment.

5.3.3 Maintenance risks

According to The Free Dictionary (2013), the word 'maintain' can be defined as:

- to provide for, or to support; or
- to keep in existence, or to sustain.

From a database perspective, maintenance can include all the tasks that are related to supporting and sustaining the database, in order for the database to perform as the business intends. These tasks are performed by the IT employees, or by the company supporting the database. Table 5.3 below discusses some of the maintenance tasks that can be performed when using Cassandra (Datastax, 2013b), and the business imperatives on which they can impact.

Table 5.3: Cassandra's operational IT maintenance risks

Task	Detail of task	Business imperatives impacted
Monitoring	Monitoring a Cassandra cluster can be done using any of the following tools: • DataStax OpsCenter management console; • Cassandra nodetool utility; and • Jconsole. Using these tools, the performance of the database can be monitored, and certain administrative tasks can be performed.	Hyper performance: Monitoring should be regularly performed and appropriate actions taken in order for the business to meet its performance requirements.
Providing backup	The data within Cassandra can be backed up by means of taking a 'snapshot'. The process, which is performed per node, entails storing all on-disk data files. The data can then be moved offsite, if so required, and the cluster can be restored with the 'snapshot'.	Zero downtime: Cassandra is designed to be a highly available database, but, in the unlikely event that all the data within the cluster requires restoring, backups should be available.

Task	Detail of task	Business imperatives impacted
Repairing nodes	Nodes should be repaired during the following circumstances: • during normal operations as routine cluster maintenance; • in order to recover a node after it has failed; • where nodes containing data are not frequently accessed; and • the updating of a node after being offline.	Hyper performance: Nodes that are not online tend to impact on performance, due to the linear scalability. Zero downtime: The failure of large numbers of nodes can impact on the availability of the database.
Upgrading	Before upgrading to a newer version of Cassandra, the changes that might impact on the upgrade should first be understood, and the appropriate corrective steps taken. Following the recommended steps and taking suitable measures (for example, to provide backup before upgrading) in order to restore the data should something go wrong is advisable.	Zero downtime: Upgrading might impact on the availability of the database.

Task	Detail of task	Business imperatives impacted
Handling of schema disagreements	Cassandra provides the ability to resolve schema disagreements automatically, but, in the event that such disagreements do occur, they can be resolved by making use of a command within the command line interface.	The handling of such disagreements is unlikely to have a significant impact.
Compaction	Compaction happens automatically by means of combining multiple data files, in order to improve performance.	Hyper performance: Compaction can impact on performance if the compaction parameters are not properly managed.
Adding to, or removing nodes from, the database cluster	The addition of new nodes includes installing Cassandra on the node, and assigning the relevant properties. To remove a node, it should first be identified, in order to be able to run the node removal command.	Hyper performance: Adding nodes increases the performance of the database. The right number of nodes should be added to the database so that performance requirements are met.

Properly maintaining the database can help to ensure that the database operates as intended, and therefore should go a long way towards meeting the needs and the

business imperatives of the business. The different maintenance tasks have different business impacts, and therefore management should determine which tasks they consider to pose the biggest risks, and ensure they are properly addressed.

5.4 Conclusion

In this chapter, the incremental operational IT risks related to the use of NoSQL databases, as well as how they can impact the business imperatives, were discussed. The operational risks, in contrast to the strategic risks, are more technical in nature, and relate to the operations of the IT department.

As discussed during this chapter, the operational IT planning risks have a direct impact on the IT strategic risks. The IT design risks, as they relate to the operations of IT, occur due to impacts that the strategic IT skills risks have thereon. Therefore, the IT planning and design risks will undoubtedly pose significant challenges for management with regards alignment of IT and strategic goals.

During the discussion of Cassandra-specific risks, it was determined the configuration and software build as including more risks than setting up and hardware build parameters. This is due to the lack of standards and best practises as well as the specialist skills requirement. Therefore, the risks related to configuration and software build have a broad impact on the business imperatives. However, the operational IT maintenance risks proved to have a more direct impact on specific business imperatives.

Therefore, in order to reduce the gap between IT and the business, and achieve better alignment of IT and strategic goals, these risks should be appropriately addressed.

6. COBIT mapping

6.1 Introduction

COBIT is a widely accepted framework that assists businesses with the governance and management of IT (COBIT 5, 2012). COBIT, which is generic, caters for all businesses by providing assistance with aligning the IT and business objectives. The COBIT framework was selected due to its business focus, and due to its wide acceptance as an academic research tool.

COBIT 5 presents 37 processes categorised in five domains, which consist of the following:

- Evaluate, Direct and Monitor (EDM);
- Align, Plan and Organise (APO);
- Build, Acquire and Implement (BAI);
- Deliver, Service and Support (DSS); and
- Monitor, Evaluate and Assess (MEA).

In this chapter, the strategic and operational IT risks, as were discussed in Chapters 4 and 5, are mapped to the processes of COBIT. Thereafter, the COBIT processes that are impacted on the most by the risks are identified. The goal of this mapping exercise is to display how the strategic and operational IT risks will impact the governance and management of IT within a business that implements a NoSQL database. Therefore, this can be used as a tool, which can assist the business, in order to reduce the gap that exists between IT and the business.

6.2 Mapping of risks to COBIT

The strategic and the operational risks that were discussed in Chapters 4 and 5 are mapped to the relevant processes of the COBIT framework in Table 6.1 below. The risks involved were only mapped to the relevant process if the impact was significant.

Table 6.1: Mapping of strategic and operational IT risks to COBIT

COBIT 5 PROCESS			Stategic risks					Operational risks				
			Acquisition and vendor sustainability risk	Obsolescence and retrofit risk	Alignment risk	Skills risk	Interoperability risk	Plan	Design	Setup/Configure/Build	Operate	Maintain
Evaluate, Direct and Monitor	EDM01	Ensure Governance Framework Setting and Maintenance			X							
	EDM02	Ensure Benefits Delivery			X							
	EDM03	Ensure Risk Optimisation	X	X		X	X					
	EDM04	Ensure Resource Optimisation			X							
	EDM05	Ensure Stakeholder Transparency			X							
Align, Plan and Organise	APO01	Manage the IT Management Framework			X							
	APO02	Manage Strategy	X	X		X	X					
	APO03	Manage Enterprise Architecture	X	X		X	X					
	APO04	Manage Innovation										
	APO05	Manage Portfolio										
	APO06	Manage Budget and Costs										
	APO07	Manage Human Resources				X						
	APO08	Manage Relationships										
	APO09	Manage Service Agreements	X									
	APO10	Manage Suppliers	X									
	APO11	Manage Quality										
	APO12	Manage Risk	X	X		X	X					
	APO13	Manage Security										

			Stategic risks					Operational risks				
			Acquisition and vendor sustainability risk	Obsolescence and retrofit risk	Alignment risk	Skills risk	Interoperability risk	Plan	Design	Setup/Configure/Build	Operate	Maintain
COBIT 5 PROCESS												
Build, Acquire and Implement	BAI01	Manage Programmes and Projects						X	X			
	BAI02	Manage Requirements Definition					X	X				
	BAI03	Manage Solutions Identification and Build						X	X	X		
	BAI04	Manage Availability and Capacity				X				X	X	X
	BAI05	Manage Organisational Change Enablement			X							
	BAI06	Manage Changes								X		
	BAI07	Manage Change Acceptance and Transitioning										
	BAI08	Manage Knowledge				X		X	X	X		X
	BAI09	Manage Assets										
	BAI10	Manage Configuration								X		
Deliver, Service and Support	DSS01	Manage Operations								X		
	DSS02	Manage Service Requests and Incidents								X		
	DSS03	Manage Problems										X
	DSS04	Manage Continuity										X
	DSS05	Manage Security Services										
	DSS06	Manage Business									X	

			Stategic risks					Operational risks				
COBIT 5 PROCESS			Acquisition and vendor sustainability risk	Obsolescence and retrofit risk	Alignment risk	Skills risk	Interoperability risk	Plan	Design	Setup/Configure/Build	Operate	Maintain
Monitor, Evaluate and Assess	BAI01	Manage Programmes and Projects		X		X	X	X	X			
	BAI02	Manage Requirements Definition			X			X		X		
	BAI03	Manage Solutions Identification and Build		X					X	X		

6.2 Mapping results

The mapping, as presented in Table 6.1, shows that most of the COBIT processes will be significantly impacted by the risks relating to the use of a NoSQL database. Therefore, it is clear that the implementation of a NoSQL database will have serious implications with regards the governance and management of IT.

The COBIT processes that are impacted on most frequently by the mapping are illustrated below. Figure 6.1 excludes those processes that are only impacted on once.

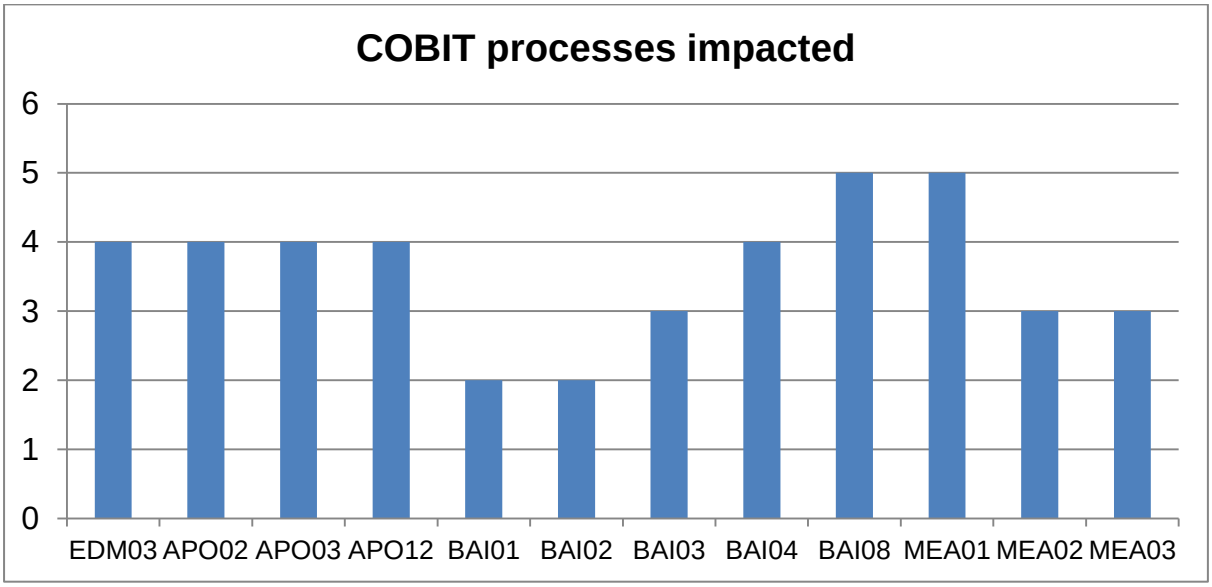


Figure 6.1: COBIT processes impacted by the strategic and operational IT risks of NoSQL

The COBIT processes, as illustrated by Figure 6.1, can therefore be considered to reserve the highest risk for the governance and management of IT. Accordingly, management should ensure that these processes are efficiently managed to appropriately respond to the NoSQL strategic and operational IT risks. By focusing on these processes, a business is in a position to better align its IT and business goals than it might otherwise be.

7. Conclusion

NoSQL databases are still immature compared to relational databases (Adrian, 2012), and have not yet achieved widespread adoption (Indrawan-Santiago, 2012). The risks, as they relate to the adoption of NoSQL databases by business, have significant business impacts that need to be considered by management.

The author of this research assignment has attempted to provide more information than was previously available regarding the business risks that are related to NoSQL databases, in order for a business to reduce the gap that exists between IT and the business. This was performed by first identifying the drivers, common characteristics and benefits of NoSQL databases. The drivers of NoSQL databases have had many influences on the common characteristics that they occupy. Also, the benefits that NoSQL databases provide stem from these drivers.

The business imperatives, that is applicable to a business that is considering the implementation of a NoSQL database, were identified based on these drivers, characteristics and benefits. The identified business imperatives can serve as the starting point for a business in order to align their IT and business goals, thereby reducing the gap between IT and the business.

Thereafter, the strategic IT risks of NoSQL databases where discussed. The strategic IT risks directly impacts the business imperatives and therefore also the alignment of IT and the business. Management should therefore incorporate these risks within their strategic IT plan to ensure they are appropriately addressed and mitigated.

The operational IT risks and how they impact the strategic IT risks and business imperatives, was discussed during Chapter 5. Correspondingly, the operational IT risks also have an impact on the alignment of IT and the business; therefore, these risks should be thoroughly considered by managed.

The strategic and operational IT risks, which were discussed in Chapters 4 and 5, were mapped to the processes of COBIT, a framework that provide assistance with aligning the IT and business objectives. The risks was determined to have significant impacts on the alignment of IT and the business, and, therefore also the IT gap.

It has therefore become clear throughout this research, that, in order to effectively align the IT and business goals, to reduce the gap that exists between IT and the business, management should consider the many risks that are related to NoSQL databases. This was performed by first defining the relevant business imperatives, followed by determining the strategic and operational IT risks, as well as how these risks can impact the identified business imperatives. This research paper can therefore serve as a framework to assist businesses in order to improve the alignment of their IT and business goals.

References

Adrian, M. 2012. *Who's Who in NoSQL DBMS's.* Gartner. Research report. 7 June 2012.

Anand, S. 2010. *Paper: Netflix's transition to high-availability storage systems*. High Scalability: Building bigger, faster, more reliable websites. [Online]. Available: http://highscalability.com/blog/2010/10/22/paper-netflixs-transition-to-high-availability-storage-syste.html [2013, October 8].

Andlinger, P. & Gelbmann, M. 2013.*Cassandra advances into the top 10 most popular database engines.* [Online]. Available: http://db-engines.com/en/blog_post/20 [2013, September 22].

Babcock, C. 2010. What's so great about NoSQL? *Information Week*, 1289:26.

Bartholomew, D. 2010. SQL vs. NoSQL. *Linux Journal,* 2010(195):4.

Bonnet, L., Laurent, A., Sala, M., Laurent, B. and Sicard, N. 2011. Reduce, You Say: What Nosql can do for Data Aggregation and Bi in Large Repositories, in *22nd International Workshop on Database and Expert Systems Applications (DEXA)*. 29 August to 2 September, Toulouse, France. IEEE Computer Society: 483-488.

Boshoff, W. 2012. Business imperatives. Masters in Commerce (Computer Auditing). [Lecture slides]. Stellenbosch: Stellenbosch University.

Boshoff, W. 2013.Technology gap. Masters in Commerce (Computer Auditing). [Lecture slides.] Stellenbosch: Stellenbosch University.

Bosworth, B. 2013. DataStax CEO: Let's clear the air about NoSQL and ACID. *InfoWorld.Com*, 24 July. [Online]. Available: http://www.infoworld.com/t/nosql/datastax-ceo-lets-clear-the-air-about-nosql-and-acid-223329 [2013, July 27].

Brewer, E.A. 2000. Towards robust distributed systems, in *Proceedings of the Nineteenth Annual ACM Symposium on Principles of Distributed Computing*. 16-19 July, Portland, Oregon, United States. PODC '00. New York: ACM: 7.

Brooks, J. 2011. Does NoSQL matter to your company? *Eweek,* 28(15):28-29.

Cattell, R. 2011. Scalable SQL and NoSQL Data Stores. *ACM SIGMOD Record,* 39(4):12-27.

Chamberlin, D.D. & Boyce, R.F. 1974. SEQUEL: A Structured English Query Language, in Proceedings of the 1974 ACM SIGFIDET (now SIGMOD) Workshop on Data Description, Access and Control. New York: ACM: 249-264.

Chang, F., Dean, J., Ghemawat, S., Hsieh, W., Wallach, D., Burrows, M., Chandra, T., Fikes, A., Gruber, R. 2008. Bigtable: A distributed storage system for structured data. *ACM Transactions on Computer Systems (TOCS),* 26(2):4.

COBIT 5. 2012. IT Governance Institute 2012. [Online]. Available: http://www.isaca.org/cobit/Pages/CobitFramework.aspx [2013, September 28].

Couchbase. 2013. *Why NoSQL? Three trends disrupting the database status quo.* [Online]. Available: http://www.couchbase.com/sites/default/files/uploads/all/whitepapers/NoSQL-Whitepaper.pdf [2013, October 27].

Codd, E. 1970. A relational model of data for large shared data banks. *Communications of the ACM,* 13(6):377-387.

Datastax. 2013a. *NoSQL in the enterprise: A guide for technology leaders and decision-makers.* [Online]. Available: http://www.datastax.com/wp-content/uploads/2011/09/WP-DataStax-NoSQL.pdf [2013, September 1].

Datastax. 2013b. *Datastax: Apache cassandra 2.0.* [Online]. Available: http://www.datastax.com/documentation/cassandra/2.0/webhelp/index.html#cassandra/gettingStartedCassandraIntro.html#concept_ds_k2h_ths_jl [2013, September 22].

DeCandia, G., Hastorun, D., Jampani, M., Kakulapati, G., Lakshman, A., Pilchin, A., Sivasubramanian, S., Vosshall, P & Vogels, W. 2007. Dynamo: Amazon's highly available key-value store. *ACM Symposium on Operating Systems Principles*, 14(17): 205-220.

DB-Engines Ranking. 2013. [Online]. Available: http://db-engines.com/en/ranking [2013, September 28].

Evans, E. 2009. *NoSQL: What's in a name?* [Online]. Available: http://blog.sym-link.com/2009/10/30/nosql_whats_in_a_name.html [2013, August 26].

Floratou, A., Teletia, N., DeWitt, D.J., Patel, J.M. & Zhang, D. 2012. Can the Elephants Handle the NoSQL Onslaught? In *proceedings of the VLDB Endowment,* 5(12): 1712-23.

Gantz, J. & Reinsel, D. 2012. The digital universe in 2020: Big data, bigger digital shadows, and biggest growth in the Far East. *IDC iView: IDC Analyze the Future.*

Gilbert, S. & Lynch, N. 2002. Brewer's Conjecture and the Feasibility of Consistent, Available, Partition-Tolerant Web Services. *ACM SIGACT News,* 33(2): 51-9.

Group Partners. 2008. [Online]. Available: http://www.grouppartnerswiki.net/index.php?title=Business_Imperatives [2013, October 27].

Hecht, R. and Jablonski, S. 2011. NoSQL Evaluation: A use Case Oriented Survey, in *International Conference on Cloud and Service Computing (CSC)*. 12–14 December, Hong Kong, China. IEEE Computer Society: 336-341.

Indrawan-Santiago, M. 2012. Database research: Are we at a crossroad? Reflection on NoSQL, in 15th International Conference on Network-Based Information Systems (NBiS). 26-28 September**:** 45-51.

IODSA (Institute of Directors Southern Africa). 2009. *King Report on corporate governance for South Africa (King III).* Johannesburg: IODSA.

Lai, E. 2009. No to SQL? anti-database movement gains steam. *Computerworld Software*, 1, July.

Lakshman, A. & Malik, P. 2009. *Cassandra: Structured storage system over a P2P network.* [Online]. Available: http://static.last.fm/johan/nosql-20090611/cassandra_nosql.pdf [2013, October 8].

Leavitt, N. 2010. Will NoSQL databases live up to their promise? *Computer*, 43(2):12-14.

Lerner, R.M. 2010. NoSQL? I'd prefer SomeSQL. *Linux Journal,* (192):20-23.

Linden, G. 2006. Make Data Useful.

Mayer, M. 2009. In Search of a Better, Faster, Stronger Web.

Mohan, C. 2013. History repeats itself: Sensible and NonsenSQL aspects of the NoSQL hoopla, in Proceedings of the 16th International Conference on Extending Database Technology, New York: ACM: 11-16.

MongoDB. 2013. *Top 5 Considerations When Evaluating NoSQL databases.* [Online]. Available: http://info.10gen.com/rs/10gen/images/10gen_Top_5_NoSQL_Considerations.pdf [2013, October 27].

Nance, C., Losser, T., Iype, R. & Harmon, G. 2013. Nosql vs RDDBMS - Why there is Room for both, in Proceedings of the Southern Association for Information Systems Conference. 8-9 March, Savannah, Georgia, United States.

North, K. 2010. The NoSQL alternative. *Information Week*, 1268:33.

Okman, L., Gal-Oz, N., Gonen, Y., Gudes, E. and Abramov, J. 2011. Security Issues in Nosql Databases, in *10th International Conference on Trust, Security and Privacy in Computing and Communications (TrustCom)*. 16-18 November, Changsa, Hunan Province, China. IEEE Computer Society: 541-547.

Oliver, A. 2012. The time for NoSQL standards is now. *InfoWorld.Com, 2013 June 7.* [Online]. Available: http://www.infoworld.com/d/application-development/the-time-nosql-standards-now-205109 [2013, July 27].

Padhy, R.P., Patra, M.R. & Satapathy, S.C. 2011. RDBMS to NoSQL: Reviewing some next-generation non-relational databases. *International Journal of Advanced Engineering Science and Technologies,* 11(1):15-30.

Pokorny, J. 2013. NoSQL Databases: A Step to Database Scalability in Web Environment. *International Journal of Web Information Systems,* 9(1):69-82.

Shalom, N. 2009a. *No to SQL?anti-database movement gains steam – my take.* [Online]. Available: http://natishalom.typepad.com/nati_shaloms_blog/2009/07/no-to-sql-anti-database-movement-gains-steam-my-take.html [2013, August 27].

Shalom, N. 2009b. *The common principles behind the NOSQL alternatives.* [Online]. Available: http://natishalom.typepad.com/nati_shaloms_blog/2009/12/the-common-principles-behind-the-nosql-alternatives.html [2013, August 23].

Stonebraker, M. 2011. Stonebraker on NoSQL and enterprises. *Communications of the ACM*, 54(8):10-11.

Stonebraker, M., Madden, S., Abadi, D.J., Harizopoulos, S., Hachem, N. & Helland, P. 2007. *The end of an architectural era: (it's time for a complete rewrite)*, in *Proceedings of the 33rd International Conference on Very Large Data Bases*.

Strauch, C. 2011. *NoSQL databases.* [Online]. Available: http://www.Christof-Strauch.de/nosqldbs.Pdf [2013, July 27].

The Free Dictionary. 2012. [Online]. Available: http://www.thefreedictionary.com [2013, September 28].

Tudorica, B.G. and Bucur, C. 2011. A Comparison between several NoSQL Databases with Comments and Notes, in *10th Roedunet International Conference (RoEduNet)*. 23-25 June, Iasi, Romania. IEEE Computer Society: 1-5.

Webopedia. 2013. [Online]. Available: http://www.webopedia.com/TERM/D/database_management_system_DBMS.html [2013, October 27].

Wikipedia. 2013a *Schema.* [Online]. Available: http://en.wikipedia.org/wiki/Database_schema [2013, July 7].

Wikipedia. 2013b *Web 2.0.* [Online]. Available: http://en.wikipedia.org/wiki/Web_2.0 [2013, October 20].

Xiang, P., Hou, R. and Zhou, Z. 2010. Cache and Consistency in Nosql, in *3rd IEEE International Conference on Computer Science and Information Technology (ICCSIT)*. 9-11 July, Chengdu, China. IEEE Computer Society: 117-120.

Appendix A – Glossary of terms

Term	Description
ACID	Atomicity, consistency, isolation, durability - ACID is a set of properties that is used by databases to guarantee the integrity and reliability of data
BASE	Basically available, soft-state, eventually consistent - BASE is a set of properties that describes databases that trade consistency for improved availability.
CAP theorem	Consistency, availability, partition tolerance - The CAP theorem, based on Eric Brewer's (2000) ideas, and theorised by Gilbert and Lynch (2002), states that one cannot achieve all three of the identified properties within a distributed system.
DBMS	Database management system - A collection of programs that enables you to store, modify, and extract information from a database (Webopedia, 2013).
RDBMS	Relational database management system – A DMBS that is based on the mathematical concept of a relation that is implemented by relational databases (Bartholomew, 2010).
SQL	Structured query language – A query language that was designed to be used with relational databases (Bartholomew, 2010).
Unstructured data	Data that are not organised into a well-defined schema and includes data types such as documents, email, multimedia and social media
Web 2.0	Web 2.0 can be defined as the second version of the World Wide Web whereby the websites enables users to interact and collaborate with each other (Wikipedia, 2013b)