# Trends of NoSQL over SQL in the World of Big Data

Santhosh Cholleti, Deekshith Sandesari
*Department of Computer Science, The University Of Akron*
*{sc145, ds168}@zips.uakron.edu*

**Abstract**

*With the advent of Big Data, managing the huge volume, velocity and variety of data is becoming a great challenge. With the rise of social media and the availability of low-cost sensors made the organizations to collect data from everything for example from wind turbines, from farms, from vehicles etc. Most of these data is unstructured and the rate at which the data is produced is also very high. The continuous growth and need to process huge volume of data in shorter time has become a crucial task for organizations. This ushered NoSQL to deal with the data management problems. The inherently horizontal scale-out architectures, flexible natures towards schema and wide variety of data models have become the important factors to tackle big data related problems. The aim of this paper is to find the factors led in choosing NoSQL over SQL. This paper discusses about big data and the data management challenges. This paper also discusses about SQL, NoSQL and its characteristics, various categories of NoSQL data models and suitable and unsuitable scenarios for each category with some case studies.*

## 1. Introduction

**"**Big data is high-volume, high-velocity and high-variety information assets that demand cost-effective, innovative forms of information processing for enhanced insight and decision making" [3]. There is vast amount of public information available online and is growing on continuously. Also, with the rise of low-cost sensors making the organizations to collect the data from every means, for example from manufacturing plants, from farms, from mobile devices etc. This resulted in handling huge volumes of data. Most of these data is unstructured and is growing at a faster rate. These huge volume, velocity and variety of data need to be managed and need to be processed to meet the ever-growing business needs and to evolve. Relational databases, which were

well established, can be considered as a possible option to handle these data but recently NoSQL databases are came into picture. Both SQL and NoSQL have their importance within their respective areas but using right tool for the right job is very important. The aim of this paper is to find the factors led in choosing NoSQL over SQL databases and it also discuss about various data-models of NoSQL databases and the scenarios that suite these data models.

### 1.1 Motivation

Relational databases are well established and provide most reliable solutions to various kinds of problems. These databases are mainstay and are in market managing all types of data regardless of their nature. But with the advent of big data and the trends to manage and process huge volume and wide variety of data in a shorter period of time with high-scalability, high throughput and in a cost-effective way led to look towards NoSQL databases an alternative for traditional relational databases. Unlike relational, NoSQL databases are horizontally scalable and it relies on commodity hardware. The giants in the IT industry such Google, Facebook and Amazon has developed their own data stores to manage and process their business needs in a better and cost-effective way. Google has developed Bigtable [4] to store and process large volumes of data like they appear e.g. social media of cloud computing applications. Amazon has developed Dynamo [2] and Facebook has developed Cassandra [2] to address their needs to in a cost effective way. These companies are primarily looking towards high scalability and cost effective alternatives. The distributed storage built on commodity hardware can be more reliable than many existing high-end databases.

### 1.2 Yesterday's Vs Today's Needs

Over time the needs for data storage have been changed considerably. In 1970's the databases were designed for single, high-end machines [2]. Initially, the relational databases were designed for centralized deployments but not for distributed environments. Even though, we can achieve the distribution by adding few enhancements

on top of them but still a gap exists as they were not designed having distribution concepts. For example, synchronization is often not implemented efficiently but requires expensive protocols like two or three phase commits [2]. In addition, the relational databases suit well for structured data with relations and allows for dynamic queries.

In contrary to this, today, most of the giant companies such as Google, Facebook and Amazon etc. are using cheap commodity hardware that may predictably fail. They are handling such failure through applications. And in today's web world, most of the data is unstructured and prepared statements or stored procedures instead of dynamic queries.

The remainder of this paper will discuss about NoSQL, how NoSQL handles big-data problems, about scalability, different data-models.

## 2. NoSQL Databases

NoSQL stands for "Not Only SQL". NoSQL databases are non-relational databases, and are different from traditional relational databases. Unlike relational databases, NoSQL databases are based on BASE properties. The successful implementation of Google's Bigtable and Amazon's Dynamo led as an inspiration for many NoSQL databases.

### 2.1 Characteristics of NoSQL

Some of the characteristics of NoSQL databases are:

#### 2.1.1 Eventual Consistency

NoSQL databases follow BASE properties, the acronym of BASE is

*Basically Available*: The database system always seems to work.

*Soft State*: The database does not have to consistent all the time.

*Eventual Consistency: "*In a steady state, the system will eventually return the last written value" [2]. NoSQL supports massive concurrent insert and read operations compared to

updates. With the partitioning of data in place, while updating or deleting the data, there exists some latency in achieving consistency across all the nodes.

### 2.1.2 Schema-free

Relational databases operate on fixed schema, and managing schema changes in relational databases is to be done carefully and through strict software testing. Unlike relational databases, NoSQL databases have no constraints on its schema. NoSQL databases provide flexibility of adding fields to database records, even without defining the changes in the schema first [6]. This provides ease in dealing with unstructured data, where there is no strict schema.

### 2.1.3 Horizontal Scalability

In order to address the disk size limits, recovery from disk failures, and to replicate the data to enable high availability, distributed data storage is important. This scenario mandates Distributed storage to handle Big Data applications.

A RDBMS big data application need to join the data that resides on so many tables and the processing time of the query depends on how fast the data has been moved from the disk to the primary storage. Partitioned nature of data and with the presence of exclusive locks slows down the update statements. Coordinating the partitioned data, securing and releasing write locks, and maintaining the ACID compliance slows down the transaction throughput. In RDBMS, this problem is addressed using vertical scalability. Vertical Scalability is a process of adding more memory, CPU's to the existing machines. However, vertical scaling is limited to a single machine and reaches to a saturation point.

In contrast to Relational databases, NoSQL databases are horizontally scalable; it refers to adding more servers to the existing resource pool. The horizontal scaling is based on partitioning the data on to several servers; this process is referred to as sharding [3]. Some of the NoSQL systems automatically perform the sharding. The ACID non-compliance nature of NoSQL systems simplifies the burden in achieving the horizontal

scalability in NoSQL systems. In addition, only read and insert operations dominate in NoSQL databases, whereas delete and update operations fade into insignificance in terms of volume [5]. NoSQL systems attain horizontal scalability by delegating two-phase commit that is required for transaction implementation to applications [5].

### 2.1.4 Low-Cost

Most of the NoSQL systems are built using low-cost commodity hardware, instead of high-end databases. These databases help in building shared-nothing architecture, in which nodes share neither memory nor disk. Each node is independent of another node. The existence of shared-nothing architecture further ease in achieving the linear scalability as every node is independent of another. In addition, the NoSQL databases are open-source which further reduces the costs. NoSQL systems have turned into "cost-effective scalable databases".

Scalability can be considered as one main reason to use NoSQL databases. If the application needs to handle billions of queries per second, the only possible way to attain it is by adding more nodes. With cheap hardware and open-source databases it is easy to use NoSQL databases.

## 2.2 Spotify Case Study

Spotify provides streaming music and has over 40 million active users and is increasing further. To accommodate the growing users and to provide a better services without any interruption, Spotify need a database technology that can address the growing needs without performance and availability issues [8].

Initially, Spotify used a postgreSQL shop, which is a relation database. But with the growing business demands, the management realized that relational database systems couldn't manage with their scalability, availability and performance requirements [8]. "*After we had scaled up to one or two million users we started to experience some scalability problems with certain services*," said Axel Liljencrantz Backend Engineer at Spotify. "*Once you hit multiple data centers, streaming replication in postgreSQL*

*doesn't really work that well for high write volumes and so on because of its limiting architecture.*" They are in need of highly scalable solution that was highly available and could support multiple data centers [8].

To meet their business needs they have migrated from PostgreSQL to Cassandra (NoSQL database). And they have successfully achieved the scalability and availability needs. Today, Spotify has deployed over 500 Cassandra nodes across 4000 servers in 4 data centers. It has 40 million monthly active users, generated over 1.5 billion playlists created and managed in real time. It can handle more than 40,000 requests/second without latency [8].

## 3. Evolution of Types of Databases

Due to growth of unstructured data such as images videos, tweets, posts, text messages, the need for flexibility and the need to handle specific type of data for specific business needs resulted in the emergence of Schema less specialized databases. For example relational databases are very rigid in structure, if there was a need to change or add new columns to the database, then it requires downtime that can impact system availability costing money and time. Hence NoSQL Databases that are suitable for specialized tasks, where relational databases are not suitable. Some of the popular NoSQL databases are:

1. Key Value based
2. Graph based
3. Column based
4. Document Based

### 3.1. Key Value Database

A key value database is a simple database, which consists of key-value pairs where a key is string that can retrieve any type of BLOB data called value. Some of the popular key value databases in the market are Dynamo DB, Azure Table Storage, Riak, Redis, Aerospike, Foundation DB and Voldemort.

Key value databases have no query language so the data is handled using only three statements GET, PUT and DELETE. The benefits of not specifying the data type of value are that we can store any type of data. Figure 3.1.1 shows that the data in the value can be any type of data such as a webpage, XML file or a string.


**Figure 3.1.1 Key value data store [1]**

### 3.1.1 Suitable scenario

Key value databases are best suitable in the following scenarios:

1) *Storing session information:* Using session id as a key all the session related data could be stored in the value column by a single PUT statement or can be retrieved using single GET statement.
2) *User profiles, Product profiles:* Using user id as a key entire user related data could be stored in the value column.
3) *Shopping cart data:* All the products added to the shopping cart can be made available by storing the product information related to a user using user id as a key.

### 3.1.2 Unsuitable scenario

Key value databases are unsuitable for the following scenarios:

1) *Relations among data:* If relationships exist among data, then key value stores are not the best way to store the data.
2) *Query by data:* Data can be queried only by key, but we cannot query the data based on something present in the value.

### 3.1.3 AdRoll Dynamo DB use case

AdRoll is a global advertising company spread across 100 countries and targets over 10,000 advertisers. The company had the need to personalize advertisements based on visitor's browsing behavior. Using key-value store such as Dynamo DB, AdRoll could efficiently serve about 50 Billion Ad Impressions daily [9].

## 3.2 Graph Database

A graph database unlike a key value database contains three data fields such as nodes, their properties and relationships among the nodes. Figure 3.2.1 shows Node-relationship node structure of a graph database. Some of the popular graph databases in the market are Neo4J, Infinite Graph, Flock DB, Graph Base and Fallen8.

Graph databases use graph query language to query the data, to show not only the nearest nodes but also helps in looking deep into the network to find patterns among the data nodes. For example using relational data if we want to find the list of friends who will likely buy us a drink, we might need to join two or more tables which may increase the complexity and requires more processing ability. But if we use a graph store we get the result of the query using the property of relationship.

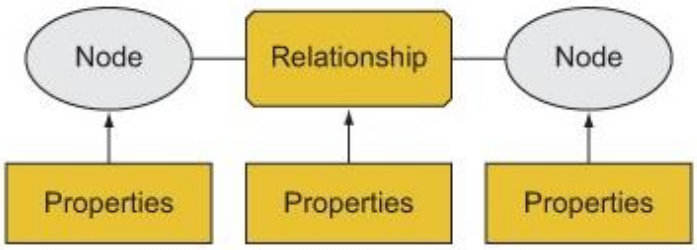


**Figure 3.2.1 Node-Relationship-Node structure of a graph database [1]**

### 3.2.1 Suitable scenario

Graph databases are best suitable in the following scenarios:

1) *Connected Data:* Graph databases can efficiently handle any link-rich domain data such as social networks, spatial data, e-commerce data or any other data that have relationships between them.

2) *Routing dispatch Services:* Considering relationship among the nodes as distance and nodes as delivery centers, delivery can be modeled to deliver the goods in an efficient way.
3) *Recommendation system:* By considering the properties of the relationships in the graph based network of a graph database, recommendation systems can make suggestion to users such as "customers who brought this product also brought these products", or populate "suggested friends in a social network website".

### 3.2.2 Unsuitable scenario

Graph databases are unsuitable in the following scenario:

1) *Modify a property of data:* If there is need to update the property of data then graph databases will not be suitable. For example if Google maps is represented as a graph database to store location name as a node and its longitude and latitude as a property of the node, changing the property of a node such as latitude will require the modification of global graph, which the database may not be able to handle.

### 3.2.3 Twitter Flock DB use case

Twitter is a popular online social network company that handles about one billion tweets per week and an average of 4, 20,000 new Twitter accounts per day [10]. With huge loads of data, Twitter used to hit the read/write MySQL limits. Since Twitter was concerned only about the relationships on the nodes such as whom we follow, it has developed Flock DB a graph database to store relationship between users. Flock DB is an open source graph database that is optimized for not only handling graph traversals but also enables fast read and writes. Using Flock DB twitter now could handle about 20k writes/sec, 100k reads/sec and store over 13 billion node relationships [10].

## 3.3 Document Database

We may recall that key value databases return the value based on the key column, that might be a path to image file or path to a document and we can operate on these values, only by keys. But document databases contain a hidden key associated to a document and we can get any document by querying the contents present in the document. Some of the

popular document databases in the market are Mongo DB, Couch DB, Couch base Server and Elastic search.

Document databases use a tree structure to represent the data, which contains a root node and combinations of sub-branches where the actual data values are stored in the leaf nodes of the tree. In a document database, unlike key value database, you could retrieve only part of the document, which eliminates the need to load the entire document into the memory. Figure 3.3.1 represents the tree structure of how data is stored in a document database. It shows how a document path is used to retrieve data of the leaf node i.e. the path to street name is People/Person [id='123']/Address/Street/StreetName/text ().

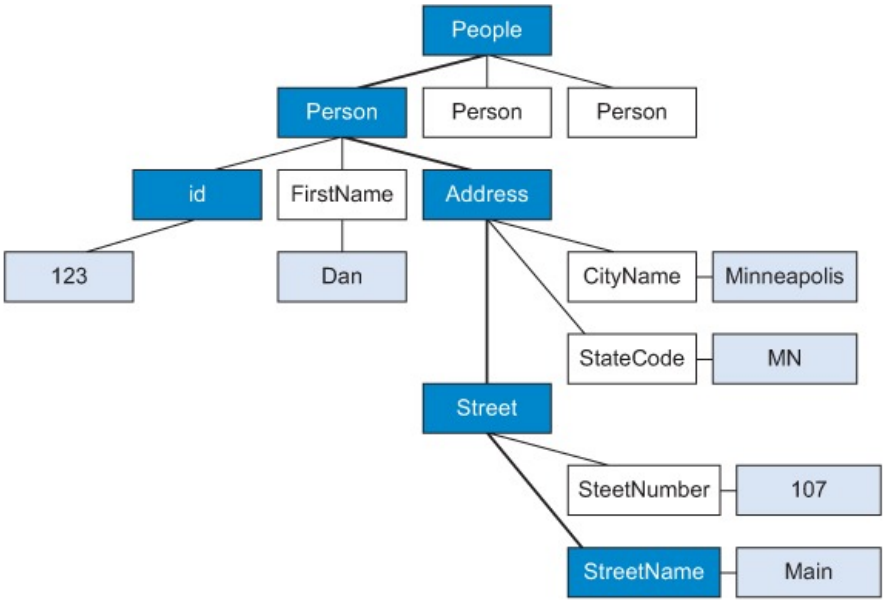


**Figure 3.3.1 Example showing Tree structure representation of a document database [1]**

### 3.3.1 Suitable scenario

Document databases are best suitable in the following scenarios:

1) *Event Logging:* Different types of applications have different event logging needs and data logged for a particular types of event keeps changing. Document databases can be used to store and manage different types of events such as order processing event or customer logging event.

2) *Content management systems:* Schema less document databases facilitates the systems in storing posts or comments on blogging platforms and also handling different types of formats such as PDF's, JSON documents.
3) *E-commerce applications:* Document databases provide the flexibility for e-commerce applications to store all the order and product related information without the need to migrate or refractor the data.

### 3.3.2 Unsuitable scenario

Document databases are unsuitable in the following scenario:

1) *Complex transactions:* Document databases does not handle the scenarios in which the databases involve in cross-atomic operations, such as handling aggregate operations like AVG, SUM.

### 3.3.3 LinkedIn Couch base use case

Linked is a business oriented social network site that has more than 225 million users across the globe generating massive amounts of user activity [11]. There is a need to monitor, log and analyze all of the user activity. Couch base Server provides high performance and scalability that helps LinkedIn to power its webserver, ultimately delivering 400,000 operations/second with their four-node cluster, and load 16 million entries into Couch base every 5 minutes [11].

## 3.4 Column Oriented Database

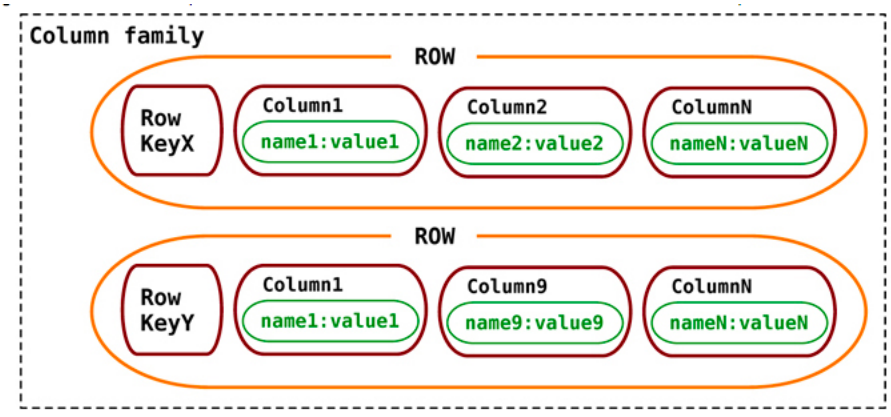


**Figure 3.4.1 Column family representation of a column database [6]**

Column oriented databases store data in column families. Each of this column family's in-turn consists of multiple rows. Each of these rows is uniquely identified by a row key and can store data in the form of columns. Figure 3.4.1 shows the representation of column-oriented data databases. Some of the popular column oriented databases in the market are Cassandra, Amazon SimpleDB, Hypertable and Accumulo.

For example, we may have groups of columns related to user info or order info. This grouping of all the columns together in a column family is useful when we need to view all the related user info together. Each of these row keys identifies the info related to a particular user. Also each row consists of different set of columns such as age, email, gender, sate etc. as shown in Figure 3.4.2. The number of columns are not same for each of these rows. This gives flexibility to the database, unlike relational databases where each table has rigid schema.

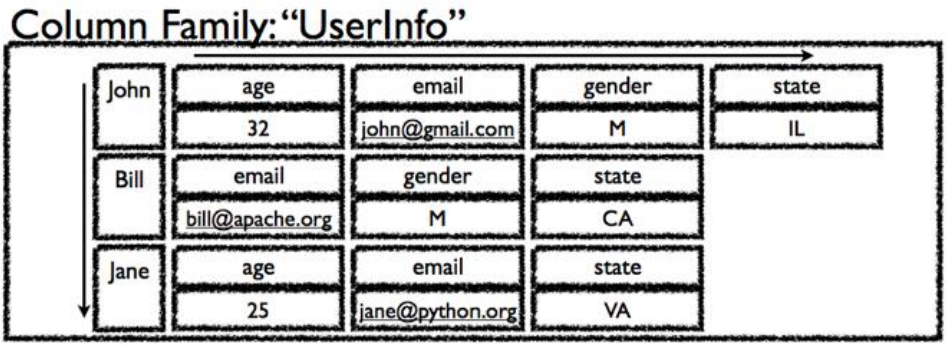


**Figure 3.4.2 Example showing data is stored in a column database [7]**

### 3.4.1 Suitable scenario

Column databases are best suitable in the following scenarios:

1) *Event Logging:* As addressed previously since event logging needs and data logged for a particular type of event keeps changing, column families can provide flexibility in storing this event data.
2) *Expiring Usage:* Column family databases provide the flexibility for the data to appear for a specific period. Data related to promotions offers and sales can be set to appear for a day or week and then deleted using expiring columns.

3) *Counters:* In web applications there might be a need to count the visits to a webpage or views to a product, so as to determine the popularity of the webpage or the product.

### 3.4.2 Unsuitable scenario

Column databases are unsuitable in the following scenario:

1) *Query flexibility:* In column based databases the queries do not provide much flexibility to aggregate the data. This type of database does not handle the transactions as well.

### 3.4.3 Netflix Cassandra use case

Netflix is the world's leading subscription service for movies and TV with over 400 million users [12]. The main challenge Netflix faced was that the existing relational databases did not have the affordable capacity to store and process immense amounts of data, with more than 2.1 billion reads and 4.3 billion writes per day. Using Apache Cassandra they could achieve throughput to more than 10 million transactions per second. Initially, Netflix had a downtime of 10 minutes for every two weeks to accommodate the changing schema of the database [12]. But the flexibility provided by Cassandra eliminated the need for downtime, providing high availability for the system.

## 4. Conclusion

The scalable nature of NoSQL databases can be considered as one major factor in choosing NoSQL databases over SQL databases. Cost effective approach, flexible schema and diversified data models also turned in as a major factors of NoSQL. Due to high volume, variety and veracity of data been generating, we have seen that since RDBMS cannot solve all the needs of an application, different NoSQL databases evolved to satisfy the business needs of different application scenarios.

Since different databases solve different types of problems, it is not necessary for the application to use a single data store for all of its needs; multiple databases can co-exist in a single application and can be used for different needs. For example in an e-commerce application, a key value database can be used store session and shopping cart data related to a user before the order is confirmed. Once if the order is confirmed the data can be

stored in Relational Databases. The entire product inventory related data can be stored in document or column based databases, which gives flexibility. Also, to recommend products to a customer when the order is placed a graph databases can be used. Thus we can say that emergence of NoSQL databases does not eliminate the need for SQL or Relational databases but both of them can co-exist to solve all of the business needs effectively in the world of big data.

## 5. References


[1] Dan McCreary, Ann Kelly, *Making Sense of NoSQL: A guide for managers and the rest of us*, Manning Publications, September 03, 2013.

[2] Prof. Walter Kriha, "NoSQL Databases", Computer Science and Media, Hochschule der Medien, Stuttgart (Stuttgart Media University), April 2009.

[3] Gartner, "Big Data", http://www.gartner.com/it-glossary/big-data/, December 2014.

[4] Chang, Fay ; Dean, Jeffrey ; Ghemawat, Sanjay ; Hsieh, Wilson C. ; Wallach, Deborah A. ; Burrows, Mike ; Chandra, Tushar ; Fikes, Andrew ; Gruber, Robert E.: "*Bigtable: A Distributed Storage System for Structured Data*". November 2006.

[5] Gudivada, V.N.; Rao, D.; Raghavan, V.V., "NoSQL Systems for Big Data Management," *Services (SERVICES), 2014 IEEE World Congress on* , vol., no., pp.190,197, June 27 2014-July 2 2014.

[6] Pramod J. Sadalage, Martin Fowler, *NoSQL Distilled: A Brief Guide to the Emerging World of Polyglot Persistence*, Addison-Wesley Professional, Indiana, August 08, 2012.

[7] Jeremiah Jordan, Talk on Databases/NoSQL - Apache Cassandra and Python, PyCon2012, California, March 2012, https://github.com/msabramo/pycon2012-notes/blob/master/apache_cassandra_and_python.rst

[8] Data Stax, Planet Cassandra, "Spotify Casestudy", http://www.datastax.com/wp-content/themes/datastax-2014-08/images/case-studies/DataStax-CS-Spotify.pdf, September 2014.

[9] Prashant Pandey, Pravin Muthukumar, "Real-Time Ad Impressions Bids Using Dynamo DB", http://aws.amazon.com/blogs/aws/real-time-ad-impression-bids-using-dynamodb/, April 02, 2013.

[10] Slain, Elf, "Introducing FlockDB", https://blog.twitter.com/2010/introducing-flockdb, May 3, 2010.

[11] Michael Kehoe, Carleton Miyamoto, “CouchBase at linkedIn”, CouchBase Connect 2014, http://info.couchbase.com/rs/northscale/images/LinkedIn_Couchbase%20case%20study.pdf, November 10, 2014.
[12] A Case study from Datastax, “Case Study: Netflix”, November, 2014 http://www.datastax.com/resources/casestudies/netflix.

## 6. Contributions

**Santhosh Cholleti (SC145)**: I worked on finding the differences between SQL and NoSQL. Studied the Background of SQL, NoSQL and the reasons for the evolution of NoSQL databases. Did some literature survey on the Google Bigtable, Amazon Dynamo and Facebook Cassandra, which stands as an inspiration for various NoSQL databases. Also did the research on the factors such as scalability, flexible schema and cost-effective nature of NoSQL.

**Deekshith Sandesari (DS168):** In this paper my contribution includes the study of factors led to the evolution of NoSQL databases apart from the factors like scalability and eventual consistency. Then I have done literature survey on different types of NoSQL databases, discussed different types of databases. After studying each type of databases, I have listed down the scenarios in which a particular type of NoSQL database is suitable and unsuitable. In addition, I have presented a use case for each type of NoSQL database. After presenting my overview I have concluded that SQL and NoSQL databases should co-exist to solve specific business needs.