Transitioning from SQL to NoSQL Data warehouses: A Need or just a trend?

Syeda Mahnur Asif
sasif@khi.iba.edu.pk

Abstract – With the advent of IoT technologies, more and more connected services and personalized features use user data that has become key to providing insights to consumer needs. This data, being exponentially generated must be stored for analysis somewhere. Traditional data warehouses with their relational databases and structure are no longer up to the task of storing such variable data, nor are they easily scalable. In this research paper a case study of an organization taking a leap from SQL to a NoSQL based data warehouse is presented and the factors that influenced this decision are outlined.

Index Terms – SQL, NoSQL, Data warehouse, Cassandra, Netflix

1 INTRODUCTION

As everything on the internet becomes increasingly connected and data dependent, this data generated by devices and user's usage behaviors becomes increasingly valuable to organizations for insights to consumer needs and better business planning. Traditional data warehouses with their relational databases and structure are now no longer up to the task of handling the high volume varied, semi- and unstructured data and store it in a way suitable for analytics. Emerging NoSQL technologies are filling this gap and more and more organizations are turning to NoSQL based data warehouses that offer greater scalability, a higher response rate and can easily process and store data of various forms.

This research paper is divided into two parts. The first part, introduces the common terminologies related to this subject. Section 2 focuses on a case study, Netflix's transition from SQL to NoSQL and highlights the factors that supported this decision, elaborating on the NoSQL technology chosen and why it was chosen. Finally, key points of how a NoSQL implementation changed the dynamics of a traditional data warehouse are explored.

1.1 Data warehouse (DW or DWH)

A data warehouse is a central repository of information that can be analyzed to make better and more informed decisions. Data in the DWH flows from a multitude of sources; relational databases, transactional systems etc. Data in the warehouse is stored through the extract, transform and load (ETL) process and is accessed through Business Intelligence tools such as SQL clients [1][4]. A widely accepted definition of DWH according to Bill Inmon is: *A data warehouse is a subject-oriented, integrated, time-variant and non-volatile collection of data in support of management's decision making process.* These four characteristics mentioned in the definition are further elaborated as follows:

Subject-oriented – DWH can be used to derive analytics specific to a particular subject or area, e.g: sales.

Integrated – Information from multiple sources and in multiple formats must be consolidated together and

be in a consistent format or completely integrated. No issues such as naming conflicts and other inconsistencies must remain [3].
Time-variant – Historical data is stored in DWHs. This is used to discover trends over time.
Non-volatile – Data once entered in the data warehouse is immutable.
An organization can be deemed suitable for data warehousing if its data meets all these four characteristics and if it has such data that needs to be analyzed.

Data warehouse's architecture is dominated by three tiers [1], bottom or data tier, consisting of the database server, data is loaded and stored here. The middle tier has the Online Analytics Processing (OLAP) Server, and the top tier which is the front-end client layer.

1.2 Not Only SQL (NoSQL)

NoSQL is a term that loosely refers to a particular type of database model, or database management system [5] that is not based on SQL, the language most commonly associated with relational databases [6]. NoSQL systems store and manage data in ways that allow for high flexibility in the structure of the data and later horizontal scalability and speed when accessing it [7]. These systems, however, do not provide the same level of consistency as the traditional relational databases provide. They also compromise on Atomicity, Consistency, Isolation and Durability (ACID) properties in favor of providing availability, partition tolerance, and speed [8]. A NoSQL database has a dynamic schema used to store unstructured data whose manipulation is done through the use of object-oriented application programming interfaces (APIs). Such databases can be of many types, as outlined below:
Document databases, pair each key with a complex data structure known as a document. Documents can contain many different key-value pairs, or key-array pairs, or even nested documents [9]. E.g: CouchDB and MongoDB.
Key-value based databases store free form values. These values can be simple integers or strings to complex JSON documents, which are then accessed through keys, e.g: Redis and Riak [7].
Wide-column stores such as those in Cassandra and HBase are optimized for queries over large datasets. These store columns of data together instead of rows [9]. Any number of columns, and therefore many different types of data, can be grouped or aggregated as needed for queries or data views [7].
Graph databases represent data as a network of entities and their relationships. Each node in the graph is a free form chunk of data. Graph stores include Neo4J and Giraph [7][9].

1.3 Structured Query Language (SQL)

SQL is a programming language designed for managing and manipulating data held in a relational database (RDBMS), or for stream processing in a relational data stream management system [10]. SQL is used to communicate with a database. It includes elements that enable data querying, data manipulation, data definition (schema creation and modification) and data access control. Although SQL is to a great extent a declarative language, it also includes procedural elements [10].

According to ANSI (American National Standards Institute), it became the standard language for relational database management systems [11] in 1987. Since then, the standard has been revised to include a larger set of features. Despite the existence of such standards, most SQL code is not completely portable among different database systems without adjustments.

There exists many different versions of the SQL language. However, to be compliant with the ANSI standard, they all support at least the major commands (such as SELECT, UPDATE, DELETE, INSERT, WHERE) in a similar manner [12].
SQL, at the time of conception, offered two main advantages over older read/write APIs like ISAM or VSAM: first, it introduced the concept of accessing many records with one single command; and second, it eliminated the need to specify *how* to reach a record, e.g. with or without an index. Today, some common relational database management systems that use SQL are: Oracle, Sybase, Microsoft SQL Server, Access, Ingres, etc [11].

SQL databases are unlike those of NoSQL and are only of one type, or table-based only. They have a predefined schema and are vertically scalable. This

means that their scalability is limited when compared to that of NoSQL databases. SQL supports complex querying and SQL databases are suitable for high transactional based applications. It is more stable and promises the atomicity as well as integrity of the data. SQL databases emphasize on ACID properties (Atomicity, Consistency, Isolation and Durability) whereas the NoSQL database follows the Brewers CAP theorem (Consistency, Availability and Partition tolerance) [13].

2 SQL TO NoSQL DATA WAREHOUSE

2.1 Case Study: Netflix's migration from Oracle to Cassandra

Netflix

Netflix is a streaming service that allows customers to watch a wide variety of award-winning TV shows, movies, documentaries, and more on thousands of internet-connected devices [14]. Netflix greatly expanded the production and distribution of both film and television series since 2012, by January 2016, Netflix services operated in more than 190 countries. As of October 2018, Netflix has 137 million total subscribers worldwide, including 58.46 million in the United States [15].

2.2 Oracle's failings and the factors that drove the transition

When Netflix came into being, it started with a more traditional MySQL database for data warehousing, storing more than 10 years of customer data and billions of ratings. However when the growth of data collected by Netflix started to increase exponentially the service started to shift towards Internet streaming. This was when Netflix ran into scalability problems. The growth of the customer base was supplemented by data collected from existing users watching more and more shows [15].
Backend for transactional datasets at this time included, Oracle, Microsoft SQL Server and MySQL. RDBMS was not flexible enough and while Oracle handled structured data it lacked a dynamic data model which was important for handling Netflix's high traffic needs.
Oracle scales up. Master slave design limits both its scalability and performance for online applications. The failure of Oracle to add more capacity online in an elastic fashion without impacting the applications was then, and still is, a major drawback.. This led to the website going down every two weeks to include changes to PL/SQL procedures, packages, schema changes and other backend database changes. In 2008, a Netflix outage based on hardware failure led to corruption of databases and thus rethinking of its infrastructure of SQL based data warehouse.

Netflix's focus became having a system that had to have these three key features: High availability, multi data center support, and could scale predicatably. Technology that fulfilled all these requirements was Cassandra [16].

2.3 Cassandra

Apache Cassandra is a type of NoSQL database management system, more specifically, it is of a wide column store type. Cassandra is free, open-source, and is designed to handle large amounts of data across many commodity servers, providing high availability with no single point of failure. Cassandra offers robust support for clusters spanning multiple data centers with asynchronous masterless replication allowing low latency operations for all clients. [17]

Main features of Cassandra are outlined ahead. Cassandra is supports a distributed architecture. All nodes in the cluster have the same role ensuring that there is no single point of failure. There is no master, and every node can service any request.

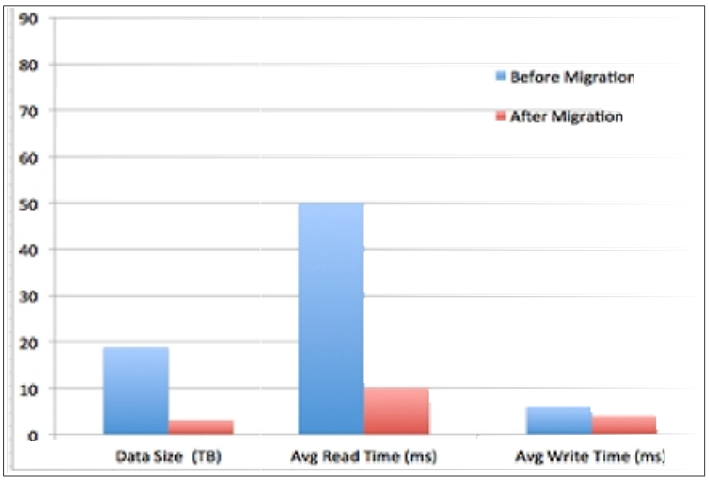
It supports replication and is specifically tailored for multiple-data center deployment, for redundancy, for fail over and disaster recovery with no downtime involved. Cassandra is scalable; read and write throughput can both increase linearly as new machines are added with the aim of no downtime or interruption to applications.

Cassandra is also fault-tolerant. Data is automatically replicated to multiple nodes for fault-tolerance. It also has Hadoop integration and supports MapReduce.

2.4 Migration to NoSQL and how it changed the dynamics of data warehousing for Netflix

For the transition from RDMS based to NoSQL data warehouse, data had to be structured differently. Firstly, de-normalization was done to get rid of the 3rd Normal form in the RDMS design. Schema in RDBMS is equivalent to key space, and table is the same as a column family in Cassandra. Reads to the system were migrated to Cassandra first. The way DW stored data changed significantly. This resulted in more data being able to be stored more efficiently, and analytics on it were easier to carry out. It introduced a dynamic data model instead of SQL's fixed schema which enabled DWH to handle high traffic data. Also Netflix found that every two weeks they'd have at least 10 minutes of downtime to put in the new schema this downtime was completely eliminated with NoSQL.[19][20]
Since at first, client to node encryption and authentication was not supported in Cassandra, this information could not be incorporated or stored where the rest of the information was also stored. Later when Cassandra v1.2 was introduced, these security features were supported and sensitive data was also migrated to it. [20]
Data would previously be retained forever in a traditional RDBMS implementation, Cassandra's Time-To-Live (TTL) feature enables data to have limited validity, and data after a period of time can be expired. This revolutionizes how data warehouse normally holds onto historical data. TTL can help specify only relevant data to be used for analysis. NoSQL is well known for being easily scalable and being able to provide read/write consistency as number of such operations increases exponentially. All this enable DWH to hold and be able to process more data, resulting in better analytics and more informed decision making.

Figure 1 (below), shows, the comparison of a few key factors before and after the migration to Cassandra.



2.5 Summary and Tabulation

Feature	NoSQL Databases	Relational Databases
Performance	High	Low
Reliability	Poor	Good
Availability	Good	Good
Consistency	Poor	Good
Data Storage	Optimized for huge data	Medium sized to large
Scalability	High	High (but more expensive)

Figure 2:

The table above highlights key differences in NoSQL and SQL or relational databases.

When NoSQL is used in a data warehousing environment, features highlighted in this figure, these attributes then extend to the data warehouse also, and how data in the data tier is accessed, its response rate, how much data is the data warehouse able to store etc.

Both SQL and NoSQL each have their advantages and disadvantages, however now with the advent of high volume of data more and more organizations are moving to implementing NoSQL based DWH for better storage capability, higher scalability and

overall performance. Security, however, of SQL based systems is better, since SQL based RDBMS adhere to the ACID properties and to the CAP theorem.
This is validated by the case study outlined above, of Netflix, and how the company resolved its scalability problems by moving from Oracle to Cassandra.

2.6 Results and Conclusion

In this research paper, definitions and key properties of terminologies such as what qualifies as a data warehouse, what is SQL and NoSQL and the different types of databases that come under it, were elaborated upon. Thus comparisons are drawn between the different technologies and how they affect the implementation of a data warehouse.
A case study, Netflix's migration from MySQL and Oracle to Cassandra is also analyzed. This highlights the factors that drive a company to transition to NoSQL, the challenges it faces before and during the transition and the aftereffects of it.
At this point in time, NoSQL technologies are still emerging thus a complete understanding of how they might affect the traditional implementation of a DWH is difficult to attain. However, major factors can still be seen, and assessed. This was attempted in the paper. In the future, as more information and more organizations adopt NoSQL how it affects the structure and performance of a data warehouse will become clearer and research of this subject can hopefully be carried out in a more definitive way.

REFERENCES

[1]https://aws.amazon.com/data-warehouse/

[2] https://www.quora.com/What-is-a-data-warehouse

[3]https://docs.oracle.com/cd/B10501_01/server.920/a96520/concept.htm

[4] https://www.sas.com/en_us/insights/data-management/data-warehouse.html

[5] https://database.guide/what-is-nosql/

[6] https://spring.io/understanding/NoSQL

[7]https://www.infoworld.com/article/3240644/nosql/what-is-nosql-nosql-databases-explained.html

[8] https://en.wikipedia.org/wiki/NoSQL

[9] https://www.mongodb.com/nosql-explained

[10] https://en.wikipedia.org/wiki/SQL

[11] http://www.sqlcourse.com/intro.html

[12] https://www.w3schools.com/sql/sql_intro.asp

[13] https://www.thegeekstuff.com/2014/01/sql-vs-nosql-db/

[14] https://help.netflix.com/en/node/412

[15] https://en.wikipedia.org/wiki/Netflix

[16] https://www.slideshare.net/acunu/cassandra-eu-2012-netflixs-cassandra-architecture-and-open-source-efforts

[17] https://en.wikipedia.org/wiki/Apache_Cassandra

[18] https://blog.panoply.io/cassandra-vs-mongodb

[19] https://medium.com/netflix-techblog/revisiting-1-million-writes-per-second-c191a84864cc

[20] https://www.datastax.com/wp-content/uploads/2011/09/CS-Netflix.pdf