International Journal of Research in Advanced Engineering and Technology
ISSN: 2455-0876
Impact Factor: RJIF 5.44
www.newengineeringjournal.in
Volume 4; Issue 2; May 2018; Page No. 58-63

Comparison of NoSQL Databases

Akhil Tandon, Dr. Abhay Kumar Agarwal
KNIT, Sultanpur, Utter Pradesh, India

Abstract
The NoSQL data models are gaining popularity, still they are far behind than the major relational data models. The reason is that the NoSQL data models are still alien for most of the organizations. Hence, the organizations neither feel knowledgeable nor confident enough to rightly decide about which of the NoSQL data model may better serve their need. Also, the organizations remain hesitant to completely shift from its reliable legacy data models, currently being used, to NoSQL data model.

Keywords: NoSQL, data model

1. Introduction
Our study helps the organizations and the individuals to select the appropriate category of database whether NoSQL or Relational Database and also know the comparative performance of these two types of database. This study helps to get familiar with the NoSQL databases as we select few databases for the study among several. In this paper we present the comparison among several NoSQL databases, which is based on their popularity and their technical features. In this paper we compare the databases based on their popularity and their technical features [19, 20, 4, 5, 6].

2. NoSQL Database Comparison
In this section comparison of NoSQL database is presented. The comparison among the NoSQL databases is defined under the categories. Popularity trends and technical features are compared and analyzed [15, 16, 18, 20].

2.1 Comparison on the Basis of Popularity- By considering the popularity trends
The various Big Data Databases are compared which are given below for the comparison of popularity we measure System
By using the following parameters

1. Number of mentions of the system on websites, measured as number of results in search engines queries. At the moment, we use Google, Bing and Yandex for this measurement. In order to count only relevant results, we are searching for <system name> together with the term database, e.g. "Oracle" and "database".
2. General interest in the system. For this measurement, we use the frequency of searches in Google Trends.
3. Frequency of technical discussions about the system. We use the number of related questions and the number of interested users on the well-known IT-related Q&A sites Stack Overflow and DBA Stack Exchange.
4. Number of job offers, in which the system is mentioned. We use the number of offers on the leading job search engines Indeed and Simply Hired.
5. Number of profiles in professional networks, in which the system is mentioned. We use the internationally most popular professional networks LinkedIn and Upwork.
6. Relevance in social networks. We count the number of Twitter tweets, in which the system is mentioned.

2.2.1 Comparison of Wide Column Stores
During studies It can be noticed that Cassandra has a bit higher popularity trends almost linearly compared to HBase. If the comparison is done among these three data models discussed in wide column category, though, BigTable is distantly far behind in popularity, but HBase is enjoys its popularity much closely with Cassandra, though, still stays at the lower end. [10, 12, 14].

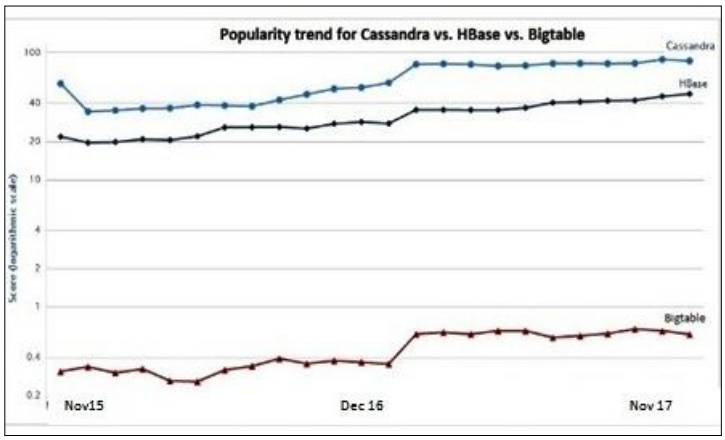

Fig 1: shows the popularity trends for Cassndra, Hbase, and big table

2.1.2 Comparison of Document Oriented Store Databases
MongoDB and CouchDB are document-oriented free and open source databases, the word "document" does not mean a word processing file or a PDF. Rather, a document is a data structure defined as a collection of named fields. JSON (JavaScript Object Notation) is currently the most widely used notation for defining documents within document-oriented databases. Under the document store category, the two most popular data models are discussed and Figure 2 depicts the popularity trends for MongoDB and CouchDB. It can be noticed that MongoDB always enjoys bit higher popularity trends compared to CouchDB.

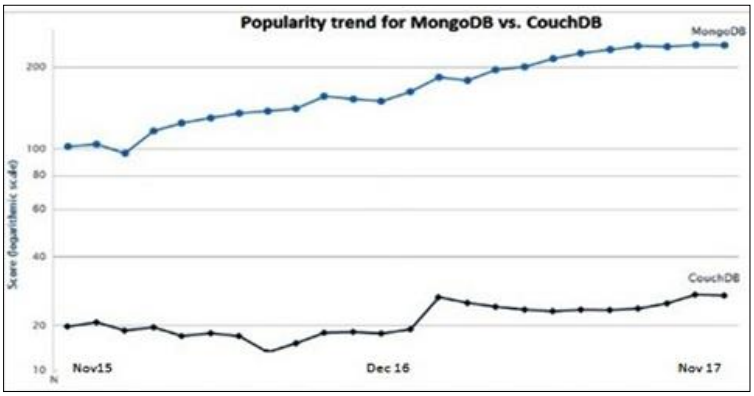

Fig 2: popularity trends for mongo DB and couchDB

2.1.3 Comparison of Graph Data Databases
We attempt to capture the popularity trends between Neo4j and OrientDB data models in Figure 3. When popularity comparison is done between only these two data models, Neo4j is being discussed far more than OrientDB in literature. Though, Neo4j continuously enjoys its up scaled popularity since its inception; but, it can be clearly noticed in the trending graph that the wider gap of popularity between these two robust data models has been abated a bit lately

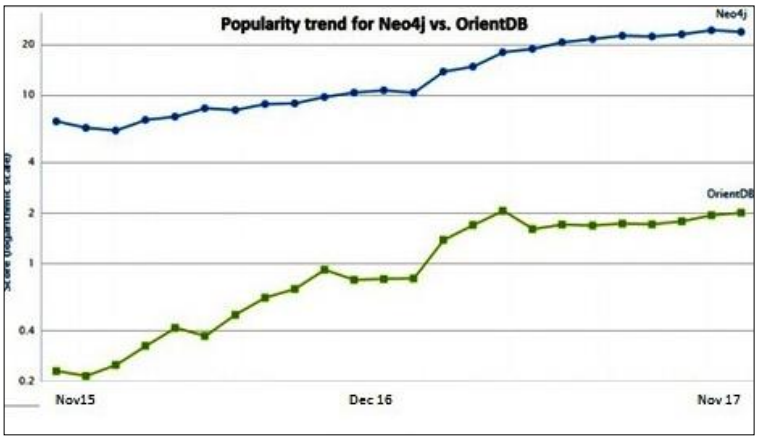

Fig 3

2.1.4 Comparison of Key Value Store Databases

We attempt to capture the popularity trends between Riak and Oracle NoSQL data models in Figure 4. As the basic foundation of Oracle NoSQL is Berkeley DB, which is very widely used (as local Big Data model) even today, we try to mention it in the discussion, wherever possible. As it is already mentioned that Berkeley DB serves as the foundation for the newly developed Oracle NoSQL, Berkeley DB still maintains its popularity over Oracle NoSQL. Figure 4 also indicates that Riak is being discussed far more comparatively in literature. Though, the popularity trend for Berkeley DB is almost linear or rather slightly decreasing, a gradual demand for Riak and Oracle

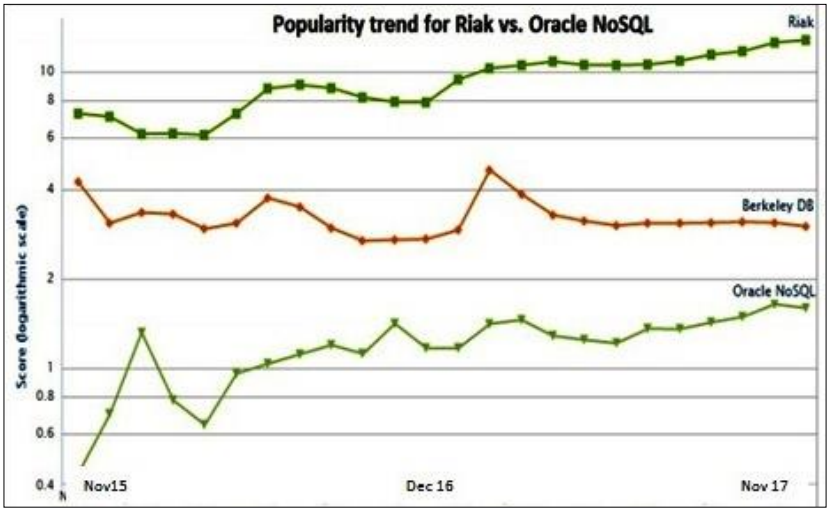

Fig 4

2.2 Comparison on the Basis Technical Features

By considering the technical features the various Big Data Databases are compared which are as follows [17, 19, 13].

2.2.1 Comparison of Wide Column Stores

Table 1 contains some of the important features of the Cassandra, HBase, and Big Table; the features are intuitive and self- descriptive. Unlike other traditional Big Data models, in HBase, MapReduce is a software programming model that is known for its efficient parallel and distributed computing capabilities. It becomes extraordinarily useful when it is convoluted with hardware of the data models on the top of them to serve as demons. As of late, the interweaving proximity of Map Reduce programming model with Apache Hadoop framework has drawn significant attention and popularity in the recent buzz of Big Data. Indexes support the efficient execution for more often used queries, especially for the read operations. An index may include a column or more and sometimes the size of an index may grow larger than the table itself, it is being created for, but eventually provides the rapid lookup and fast access of the data and this compensates for the overhead of having indexes. As Big Data models entertain multi-variable queries, the indexing is required to be in place for all the in-operation variables, promising faster access for individual variable search results that are further combined for collective response as a whole. The model should have the ability to produce smaller indexes, when there is a possibility of granularity eduction. The indexes should able to be easily portioned into sections whenever the opportunities for parallel processing arise. As the rate of data generation under the Big Data ambit is much faster, for in-situ processing, the index should be built at the same rate. In Cassandra, similar to the relational database where the first index is the primary key that ensures the uniqueness of records, the primary index, applied to a column family, serves as the index for its row keys and is maintained by each node for its associated data (Lakshman, and Malik, 2011) [13, 3]. Cassandra also provides secondary indexes on column values which can act as additional filter to the resul tset. HBase is not equipped with robust 18 indexing capability (Dimiduk and Khurana, 2012). Secondary indexes are not built-in by default, but, can be generated in other table that may require periodic updated and a Map Reduce job performs the task; another possibility is to write to the index table while writing to the data table. Through, each approach has both merits and demerits, but, all in all HBase does not aggressively support indexing. Like HBase, Big Table data model also is not a good contender of indexing (Jin et al., 2011). As, it is already stated that technically, Big Table is a map, which is sparse, distributed persistent, multi dimensional and sorted. Though, the map is indexed by a row key, column key, and a timestamp. As far as versioning (Shaughnessy, 2012) issue is concerned, in HBase, the version management of the data is assisted by the timestamps [6, 1, 10, 12, 14].

Table 1: Feature comparison of Cassandra vs. HBase vs. Big Table

Features	Cassandra	HBase	BigTable
Relational Nature	Yes	No	No
Developer	Facebook Inc.	Microsoft Powerset)	Google Inc
Written in	Java	Java	C, C++
Query language	CQL	1. Pig latin 2. HQL	APIs in C++
SQL Nature	Yes	1.No 2.Yes	No
High Availability	Yes	Yes	Yes
High Scalability	Yes	Yes	Yes
Single Point of Failure	No	Yes	N/A
Open Source	Yes	Yes	No
Versioning	Yes (no built-in)	Yes (built-in timestamp)	Yes (built-in timestamp)
Indexing	Basic Primary and Secondary	Basic (Secondary not by default)	Basic on map, supported by timestamp
Data Processing Nature	Streaming and Atomic Batches	Batch processing	Batch processing

2.2.2 Comparison of Document Oriented Store Databases

The important features of the MongoDB and CouchDB are shown in Table 3.2. The features of the data models are also compared. The importance of indexing in database has been proven very efficient since a long period of time, particularly when data size is huge. Indexing in Big Data is a challenge comparatively due to several reasons. Primarily, the volume and the velocity of the Big Data are larger to the order of magnitude comparatively; the generated index should be smaller (a fraction of original data) and equally faster as compared to the data as the search is conducted through billions or even trillions of data values in seconds. In MongoDB, indexes (Chodorow and Dirolf, 2010), which use B-tree data structure, are generated at the collection (equivalent to a table in relation databases) level. To support variety of data and queries, there are different types of index are available in MongoDB [1, 9, 11].

1. Default_id -This exists by default on _id field of collections and also serves as unique id too.
2. Single Field -This is a user defined index on a single field of a document.
3. Compound Index -This is user defined index on multiple fields.
4. Multikey Index -This is used to index a field that hold an array of value.
5. Geospatial Index -These are a very special index that MongoDB (unlike other models) provides to support spatial queries and are of two types:
   a. 2d spatial index for planer geometry queries.
   b. 2sphere spatial index for spherical geometry queries.
6. Text Index -This is the index type that helps searching for the string or text content in a collection.
7. Hashed Index-This helps indexing the hash of the values of a field.

Table 2: Feature comparison of MongoDB and CouchDB

Features	MongoDB	CouchDB
Relational Nature	No	No
Developer	10 gen	IBM
Written In	C++	Erlang
Query Language	MongoDB Adhoc Query Language	Java Script
High Availability, Open Source & Scalability	Yes	Yes
SQL Nature	No	No
Single Point of Failure	No	No
Versioning	Yes	Yes
Indexing & Secondry Index	Advance & Wide Variety	Basic Associated Map & Reduce Functions
Data Processing Nature	Batch Processing & Event Streaming	Batch Processing
Replication Methods	Master- Slave	Master- Slave & Master-Master
Map Reduce	Yes	Yes
Concurrency & Durability	Yes	Yes
Cloud based	No	No
Protocol Used	TCP/IP	HTTP
Document Format	JSON	BSON (Binary JSON)
In Memory Capabilitites	No	Yes

CouchDB on the other hand does not infuse the power of indexing directly, unlike relational database models. In contrast, the usage of index in CouchDB is assisted by map and reduces functions of the predefined Map Reduce framework (Ranger et al., 2007); any variations in the document structure of the data are well absorbed by these functions and encourage parallelism and independent computation in indexes too. The indexes created by map functions limit the window of the targeted data, the reduce queries are allowed to run to seek optimal efficiency. So, while writing the map function, building the index should also be a concern. In CouchDB, the amalgamation of a map and a reduce function is termed as a view. In sum, if we compare the indexing richness for MongoDB and CouchDB, former

distantly leads and clearly is a preferred choice for better indexing, Versioning, also termed as version control, revision control, is another important features that worth discussion. Versioning allows fetching latest or any particular version out from the tables in databases and also significantly eases the audit trailing of the data in the systems. MongoDB versions the document, through this is not its native feature but happens at application layer. The efficient way suggested is to have two different databases, where one is the application data store and other is audit trail. Though, CouchDB enjoys Multi Versioning Concurrency Control (MVCC) (Suryavanshi and Yadav, 2012; Cattell, 2011), but that does not warrant for the default document versioning support. It does implement a form of MVCC, especially to preclude the strict requirements of exclusive locking on data files while performing the write operation. Version conflicts are can be resolved at the application level by merging the conflicted data into one of the file, followed by the eradication of the rancid part [1, 9, 5].

2.2.3 Comparison of Graph Data Databases

Here Table 3 is populated with some of the important, comparable, and self-descriptive features of Neo4j and OrientDB. Neo4j does not have native versioning capacity rather it uses time-based versioning to version the data. OrientDB uses the semantic versioning (OSGi Alliance, 2010) in version assignment and tags the version in the X.Y.Z format, where X, Y, Z are integer, reflecting major, minor, and patch level values. As far as data processing is concerned, OrientDB primarily process the data in batch. Though, the data streaming is also feasible though streaming plugin [7, 8].

Table 3: Feature comparison of Neo4j and OrientDB

Features	Neo4j	OrientDB
Relational Nature	No	No
Developer	Neo Technologies Inc.	Orient Technologies
Written in	Java	Java
Query Language	Cypher	SQL
High Availability	Yes	Yes
SQL Nature	No	Yes
High Scalability	Yes	Yes
Single Point of Failure	Yes	No
Open Source	Yes	Yes
Versioning	Yes (Time Based Versioning)	Yes (Semantic Versioning)
Indexing	Legacy Indexing	SB Tree, Hash Index
Data Processing Nature	Batch processing	Batch processing Primarily

2.2.4 Comparison of Key Value Store Databases

Table 4 contains some of the very important features of Riak and Oracle NoSQL; though, most of them are self-descriptive, we briefly discuss and compare a few of them [6, 2].

Table 4: Feature comparison of Neo4j and Orient DB

Features	Riak	Oracle NoSQL
Relational Nature	No	No
Developer	Basho Technology	Oracle
Written in	Erlang	Java
Query Language	Restful API	JavaScript
High Availability	Yes	Yes
SQL Nature	No	No
High Scalability	Yes	Yes
Single Point of Failure	No	No
Open source	Yes	Yes
Versioning	Yes (Vector Clock)	Yes (Auto Incremented Integers)
Indexing	Secondary Index	Secondary Index
Data Processing Nature	Batch Processing	Batch Processing and Event Streaming

3. Conclusion

As the legacy data models which are mainly relational in nature and incapable of handling today needs and a new era of data science is commenced that gives rise to a fast emergence of wide range of Non-Relational databases. This comparison helps us to understand the list of salient features of NoSQL databases.

4. References

1. Brown MC. Getting Started with CouchDB O Reilly Media, 2011.
2. Abidi D. Overview of Oracle NoSQL Databses, 2011.
3. Lakshman malik A. The Apache Cassandra Project
4. IBM what is Big Data? Bringing Big Data to the, 2012. Enterprise www. ibm.com
5. Copper J. How entities and indexes are stored, 2009.
6. Google Inc Google Books [online] http://books.google.com
7. Online http://docs.neo4j.org
8. Garulli L, Orient DB. Orient Technologies online, 2012. http://www.orientdb.org

9. Chudorow K, Dirolf M. MongoDB. The definitive Guide. O Reilly Media, 2012.
10. Stefen Edlich List of NOSQL Databases http://nosqldatabase.org
11. Uma Bhat, Shraddha Jadhav. Moving towards Non-Relational Databases
12. Tudorica BG, Bucur C. A comparison between several NOSQL databases with comments and notes
13. Data modeling and Data Analytics: Asurvey from a Big Data Perspective by Andre Ribeiro,Afonso Silva, Alberto Rodrigues da Silva, Journal of software Engineering and Applications. 2015; 8:617-634.
14. Catell R. Scalable SQL and NoSQL Data Stores. ACM SIGMOD Record. 2011; 39:12-27. http://dx.doi.org/10.1145/1978915.1978919
15. Grolinger K, Higashino WA, Tiwari A, Capretz MAM. Data Management in Cloud Environments: NoSQL and NewSQL Data stores. Journal of Cloud Computing: Advances System and Applications. 2013; 2:22. http://dx.doi.org
16. Chang F, et al. Big table: A Distributed Storage System for Structured Data. Proceedings of the 7th Symposium an Operating Systems Design and Implementation (OSDI 06), Seattle, 6-8. 2006, 205-218.
17. GyorSodi C, Gyorodi R, Sotoc R. A comparative study of Relational and Non-Relational Database Models in a Web Based Application International Journal of Advance Computer Science & Application ISSN: 2158-107X(print) ISSN:2156-5570(online). 2015; 6(2):78-83.
18. Jatana N, Puri S, Ahuja M, Kathuria I, Gosain D. A survey and comparison of relational and non-relational database IJERT, ISSN: 2278-0181. 2012; 1(6):1-5.
19. Bulos RD, Bonsol J, Diaz R. A Lazaro, Serra V. Comparative analysis of Relational and Non-relational Database Models for simple queries in a web based application research.
20. Moniruzzaman ABM, Hossain SA. NoSQL Database: New era of Databases for Big Data Analytics, Classification, Characteristics and Comparison. International Journal of Database Theory and Application. 2013; 6:1-14.