Analysis and Evaluation of Relational and NoSQL Databases

Lokesh Gobburi, Akhilesh Bonala, Sravya Kodali
University of South Florida, CS
gobburil @mail.usf.edu, abonala@mail.usf.edu, sravyakodali@mail.usf.edu

***Abstract*— Relational database and NoSQL are competing types of data models, with the huge growth in the Internet market and the significance of Big Data made NoSQL databases surge in popularity. Choosing an appropriate database model is the biggest decision that developers must make based on the features of a given database model. We have done some analysis on properties, features of Relational Databases and NoSQL to establish which database performs better in demands of modern applications. The paper also brings out the comparison between RDBMS and NoSQL which comprises of factors like Performance, Volumes of data, Scalability and Query Language. We also present the comparison study on DML operations between MongoDB and Oracle Database. We have done an in depth analysis on which database should be chosen in which particular scenario. We will also address security issues related to Oracle and MongoDB in this paper.**

I. INTRODUCTION

In Relational Database Management System (RDBMS), data is organized into various types of database tables, records and fields. Each record in database table consists of one or more fields. In RDBMS, data is stored in the form of tables, which can be related by common fields such as database table columns. In these systems, relational operators are used to manipulate the data stored in the tables. Most of the RDBMS systems use SQL as database query language. IBM DB2, MS SQL Server, Oracle and MySQL are some of the most popular databases used today. Relational model is a specimen of record-based model. In Record based models, database is structured into fixed format records of various types. Every table in database contains records of particular type and each type of record contains a fixed number of attributes also known as fields. The columns in the database corresponds to the attributes of the record types. The most widely used model is relational data model, and a extensive majority of current database systems are based on the relational model. Dr. E.F.Cod an IBM research scientist and mathematician designed the relational model. Though many recent database systems do not comply with the Codd's definition of RDBMS, they are still considered as Relation Databases. The main focal points of Dr.Codd's design of the relational model were to further reduce data redundancy and to improve data integrity within database systems. The aim of designing a relational database is to create a collection of relationship schemes that allow us to store information without unnecessary duplication and to easily retrieve information [5].

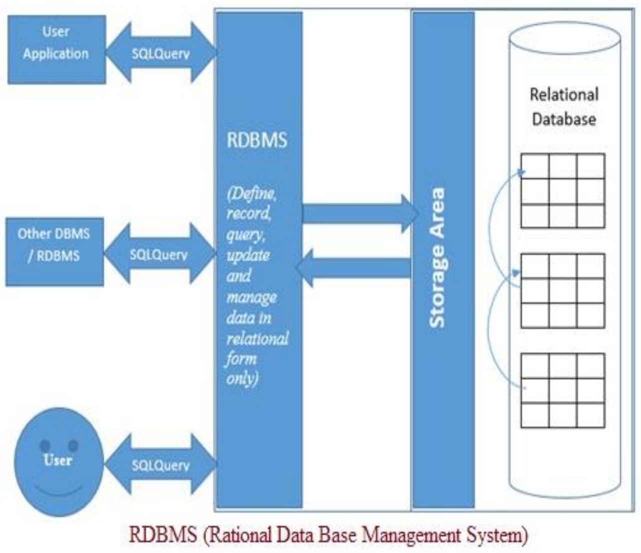

User Application
SQLQuery
Other DBMS / RDBMS
SQLQuery
User
SQLQuery
RDBMS
(Define, record, query, update and manage data in relational form only)
Storage Area
Relational Database
RDBMS (Rational Data Base Management System)

Figure 1 RDBMS [5]

On the other hand, NoSQL stands for non-SQL or non-relational database that provides a mechanism for retrieval and storage of data. The data in NoSQL is modeled such that it associates with other than the tabular relations – that are used in relational databases. Though these database models were developed during 1960s, NoSQL moniker was not obtained until their popularity rose in early 21st century [1]. The NoSQL systems can be used in wide number of applications in real-time web application involving BigData. Given their support for SQL-like query languages, NoSQL systems are also considered as “Not only SQL” [1].

NoSQL databases offers simple design, finer control on availability as well as easier horizontal scaling for thousand of machines. When compared to relational databases the data structures used in NoSQL are different from those used by default which makes some of the operations are even faster in NoSQL. Also, the suitability of NoSQL database model depends upon the problem nature. Moreover, the database structures in NoSQL are sometimes more flexible when compared to those of relational database models [4].

The use of NoSQL databases is increased significantly due to its usage which includes factors like low-level query languages, high standardized interfaces and less investments when compared to relational databases, it also offers consistency in terms of speed, partition tolerance and availability. ACID (Atomicity, Consistency, Isolation, Durability) transactions are absent in most of NoSQL stores but in few databases like MarkLogic, Google Spanner (NewSQL database) etc. have made principal to their designs.

The main advantages of working with NoSQL databases are Scalability, Availability.

- **High scalability** – For horizontal scaling, NoSQL databases use sharding – a technique where data is partitioned and placed on several machines while preserving the order of data. Additional more clusters for data handling is known as horizontal scaling whereas increasing resource capacity for existing machines is called Vertical scaling. While horizontal scaling can easily be implemented, the same cannot be said for Vertical scaling. Few examples of horizontal scaling databases include Cassandra, MongoDB etc., Because of its scalability, NoSQL can handle huge amount that is with increase in data NoSQL systems are designed to re-scale for efficient data handling [4]

- **High availability** –NoSQL databases are highly available as feature to auto replication which replicates to previous consistent state in case of data failure [4].

II. Background

One of the toughest decision that developers must make in choosing a appropriate database model (Relational Database model or a NoSQL database model).One can choose relational databases which can best option at most of the times but it is not suitable for analysis on big data and if the datasets are large. This is the primary reason which increased the adoption of NoSQL database systems in major Internet companies like Google, Yahoo, Amazon, etc. [4]

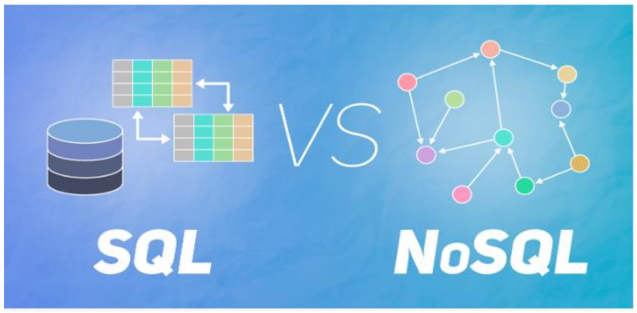


Figure 2 SQL vs NoSQL [4]

Nevertheless, choosing a database in the industry is not that easy. Both Relational and NoSQL databases have their own structure, storage methods and one can choose between SQL vs NoSQL based on the requirements of the project.

Both SQL and NoSQL databases serve the same purpose i.e. storing data, but they go about it in vastly different ways. There are multiple differences between the SQL and NoSQL databases.

For better understanding those in order to make an informed choice about the type of database required we choose MongoDB a NoSQL database management system because it is relatively new database and used in many important platforms, some includes: The Guardian, Craiglist, bit.ly etc. For example, Craiglist uses MongoDB to archive billions of records in effective way, The Guardian with the use of MongoDB discovered that they can store the documents easily, for storing user history information bit.ly uses MongoDB database management system, Oracle Database which is an object-relational database management system and It is one of the most common used and reliable relational database management systems.

Oracle DB: Oracle Database is built around relational database framework and is widely used database system developed in 1977 by Lawrence Ellision. The framework offers data objects that can be directly accessed by a front-end application or user through Structured Query Language (SQL). Another key feature of this RDBMS is that its architecture allows split between logical and physical schema is a proven advantage in grid computing (large-scale distributed computing), where the data location is transparent and irrelevant to the user. This contributes in building a modular physical structure without posing any impact to database, its users or the data. This resource sharing also helps in flexible data networks as capacity can be adjusted based on demand, without compromising on the service. Any failure can be local only as it offers networked schema of storage resources which prevents database downtime, thus proving the system more Robust [5].

MongoDB: MongoDB is a NoSQL database management system (DBMS) that supports various forms of data by using a document-oriented database model. It came into existence in the mid-2000s and it is one the numerous non-relational database technologies. MongoDB is used in big data applications and other processing jobs involving data that doesn't fit well in a rigid relational model. MongoDB architecture consists of collections and documents instead of tables and rows being used in relational database architecture. A record in MongoDB is a document can be seen as a data structure composed of field and value pairs. The basic unit of data in MongoDB is to have primary key as a unique identifier on documents. Collections contain sets of documents and function which are equivalent to relational database tables. The only restriction in MongoDB is that data in collection cannot be spread across different databases. For accommodating more data types MongoDB documents use a variant called Binary JSON (BSON) which is similar to JavaScript notation. One of the key features in MongoDB is Automatic sharding which enables data in collection to be distributed across multiple systems for horizontal scalability as data volumes and throughput requirements increase [6].

MongoDB can store any type of data and hence doesn't require any predefined schemas. This schema agonistic nature gives the user more flexibility when creating new fields in document. MongoDB databases thus simplifies the scaling when compared to relational databases.

The advantage of using these documents is the objects map to primitive data types for several programming languages. Moreover, embedded documents reduce the costs as there is less need for database joins. Horizontal scalability remains the core function of MongoDB, which is helpful for companies using Big Data applications. In addition to these features, databases can also distribute their data among several machines using sharding. Creation of zones of data using shard key is supported in newer versions of MongoDB.

III. Properties

RDBMS:

In relational databases data is stored in tables and can be retrieved at any time. It follows a hierarchical table system instead of flat files. Each table consists of rows and columns. Here data is stored in the form of objects. Along with tables we also have views in RDBMS. These are virtual tables and created using actual tables. Data will not be stored in the views. RDBMS has an interesting feature called referential integrity. Referential integrity states that a value to be entered in the foreign key field should be in the primary key field first. In RDBMS systems data must closely follow the database definition. If for some reason, there is any discrepancy then database will throw an error.

In RDBMS CRUD operations can be performed. CRUD is Create, Read, Update and Delete. RDBMS holds ACID properties. ACID is Atomic, Consistency, Isolation and Durability. Let us consider a transaction to understand it better. For a transaction to be atomic, it should be completed in full or it should be never done. Database should be in valid state after the transaction completes to satisfy the property of consistency. One transaction being performed in the database shouldn't affect any other transaction that is performed in the database. This is called isolation. Durability makes sure that the transaction is completed and committed even if the system is off for some reason [8]. All the relational databases such Oracle, Postgres, MySQL etc. support the above properties. In RDBMS we have indexes, stored procedures, packages, triggers etc.

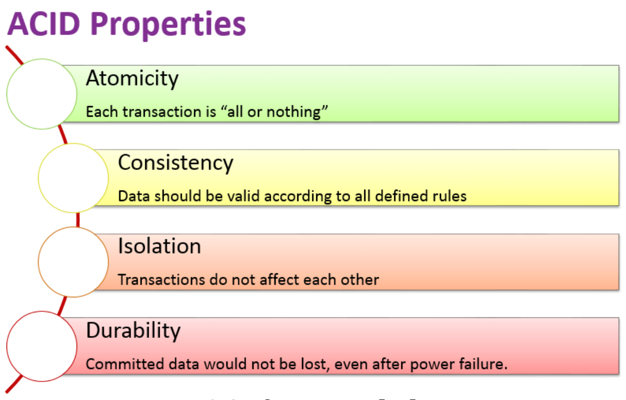

ACID Properties
Atomicity
Each transaction is "all or nothing"
Consistency
Data should be valid according to all defined rules
Isolation
Transactions do not affect each other
Durability
Committed data would not be lost, even after power failure.

Figure 3 Acid properties [14]

NOSQL:

NOSQL systems also follow the key concept of database i.e. storing and retrieving data. In some cases, storing data in flat files will do the job instead of tables which reduces the data retrieval time. But in some cases, we must use complex structures like partitioning of tables.

CAP theorem: It is also called as Brewer's theorem. According to Eric Brewer at any given time for distributed data, only two properties holds out of the three important properties Consistency, Availability and Partition tolerance [7].

- **Consistency:** It refers to properties atomic and isolation which we discussed before. Processes running concurrently will be able to see the same data. For every read operation, write or error related to the recent operation will be seen.

- **Availability:** It refers to every request will receive response. But there is no guarantee that the response has the recent write.

- **Partition tolerance:** It refers to the reliability of the system even if few requests fail. It means the system shouldn't get affected by some failures due to the network between the nodes.

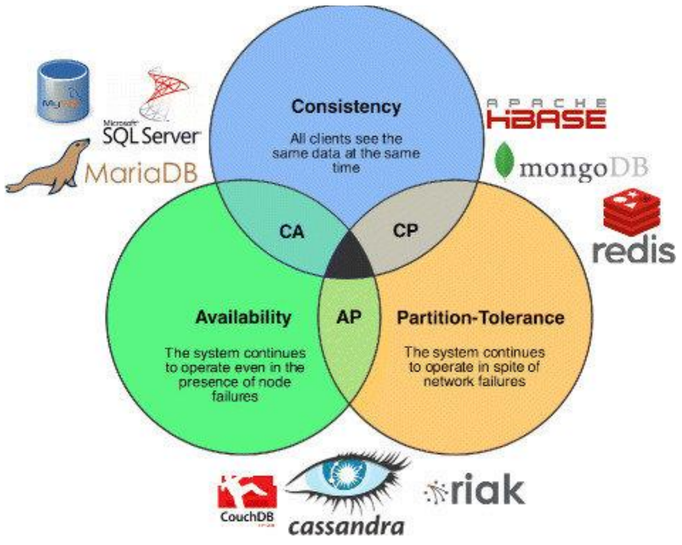

Consistency
All clients see the same data at the same time
CA
CP
AP
Availability
The system continues to operate even in the presence of node failures
Partition-Tolerance
The system continues to operate in spite of network failures
SQL Server
MariaDB
APACHE HBASE
mongoDB
redis
CouchDB
cassandra
riak

Figure 4 CAP theorem [25]

No Schema: In NOSQL systems we will not have any explicit schema [7]. So, we must follow certain rules while defining and creating data structure. We don't have to care about the internal structure, while defining and creating data structures. We can also alter the structures at any point which is a great advantage in development stage. The application is responsible for understanding and establishing logical connection between the records. It means application defines the data format and that format should be followed. If the data doesn't match with the format specified by application, then an error will be thrown. So, the data design should follow the data structure even through there is no explicit schema. That is the reason we cannot directly conclude that NOSQL system is schema less [7].

Replication: To solve the problem of data loss and to improve the system reliability, data is stored on several nodes within the network [7]. In case data is lost due to some reason then we can retrieve the data from some other node. This feature also helps in performing read operations very fast. However, we do have a drawback with this approach. It is not easy to maintain the consistency of replicas and it costs more.

Sharding: To accomplish horizontal scalability part of the original data called shard is stored in some nodes. The operations owned by the nodes can be performed in this shard [7]. However, this technique has few drawbacks. First issue is it increases the data traffic in the network. Second issue is whenever a new node is added, the probability of error increases which can be overcome by doing more replication.

So, in NOSQL we don't have ACID properties and normalization. We are left with no stored procedures and triggers. Write operations gets slower and that is why we don't have secondary indexes in NOSQL [8].

Some important features of MongoDB were listed below

- **Document Oriented:** MongoDB stores the main subject into least number of documents without splitting them into multiple relational structures.

- **Indexing:** MongoDB offers indexing for efficient search and can thus retrieve huge volumes of data in minimal time.

- **Scalability:** New machines can be deployed/added to running database. MongoDB scales horizontally by implementing sharding - partition large data into small data chunks using shard key, and evenly distribute data chunks across shards in several physical servers.

- **Replication and High Availability:** Redundant copies of data are stored in different servers and thus increase the data availability, also protecting it from hardware failures.

- **Aggregation:** Supports aggregation operations on data records. Similar to GROUPBY clause, MongoDB supports aggregation expressions like SUM, AVG, MAX, MIN etc.

IV. Data models in NoSQL

Key-Value Stores:

In Key-value stores, are system database systems that stores the values associated with a key (an index) and are searched using programmer-defined key. Each value is identified using an alpha-numeric unique key that can be synthetic or auto generated, whereas the values are uninterpreted byte arrays that can be String, BLOB (Basic Large Object), JSON (Java String Object Notation) etc. [17]. Data is stored into a collection of key-value pairs, known as associative arrays, organized into rows. A collection of Key-Value stores can by defined [19] by Ck,v = ((k1, v1),(k2,v2),....,(kn, vn))

Clients can read and write values using below key functions:

- Get(key) – Returns the value associated for key input
- Put(key,value) – Associates value with given key
- Multi-get (K1, K2,…, Kn) – Returns list of values associated with list of keys K1, K2…Kn.
- Delete (key) – Removes the key entry from data store.

In Key-value stores, relationships are handled in application logic as values are independent and isolation. Due to this simplistic architecture, they are schema-free. New values can be added at runtime without conflicts with other stored data or system availability. This simplicity makes them well suited for embedded databases, where data stored is not complex and query performance is a significant factor.

Evaluation for DB parameters:

A. *Concurrency:* Applicable on a single key and offered as optimistic writes or eventually consistent. However, optimistic writes are often not possible in highly scalable systems, because of the cost of verification that value has not been changed. Therefore, concurrency is offered either with key master where one machines owns a key or with the eventual consistency model.

B. *Queries:* There is no query language for Key-Value stores. Instead, they store, update and retrieve data using put, get and delete commands. Data retrieval path is requested directly to the object in memory or disk.

C. *Schema:* Key-Value stores do not have a schema. Instead they have two fields, a key and the value. They rely on application for parsing the data.

D. Scaling: Can implement partitioning i.e. store data on more than one node, auto recovery and replication. They can also increase the scaling by maintaining the database in RAM and can avoid locks and low-overhead server calls.

E. *Replication:* Replication of data is performed by the store itself or by the client writing to multiple servers. The replicated data is stored in the form of ring in alphabetical order as shown below.

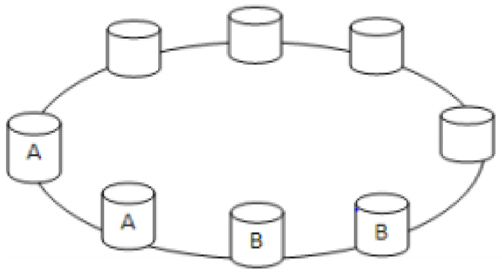


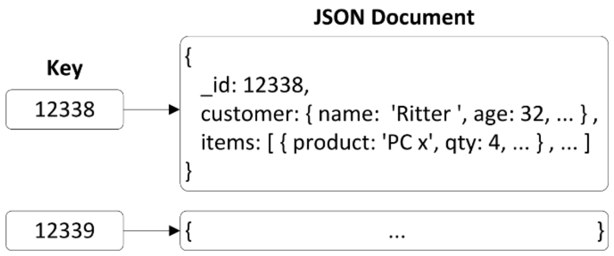
Figure 5 Ring paritioning and replication of data[17]

F. *Portability and lower operation costs:* Key value stores are portable because of their simplistic architecture. Applications can be migrated from one system to another without the overhead of re-writing the code or changing the architecture.

Therefore, for models with high performance, flexibility and scalability, we choose key value stores as they support CP of CAP theorem. Amazon Dynamo DB and Redis and popular NoSQL databases using key-value data stores. In real word applications, they are useful for storing session information, preferences, shopping cart data etc.

Document Store Databases:

Document store databases store unstructured documents such as text or semi-structured documents such as XML as records and are hierarchal in nature. The documents can also comprise nested documents, lists and scalar values. They do not have tables, rows or columns and all the information relation to a unit is stored in a document. They are similar to Key-value stores with documents comprising of keys with values. The values can be XML, JSON, and Binary JSON etc. that needs to be understood by the database. Document store databases use hashing technique where each database in the document stores pointers to its fields. Hence, they not only fetch entire document by its ID, but can also retrieve only parts of the document. They are not fixed in nature, not strongly typed and are schema-free. Document stores have a complex query language to retrieve the data and support MapReduce to process distributed data.


JSON Document
Key
12338
{
_id: 12338,
customer: { name: 'Ritter ', age: 32, ... } ,
items: [{ product: 'PC x', qty: 4, ... } , ...]
}
12339
{ ... }
Figure 6 Internal structure of document stores[18]

Evaluation for DB parameters:

A. *Concurrency:* Document stores use optimistic concurrency control i.e. they use document timestamps, multi-granularity and versioning techniques for conflict management.

B. *Queries:* As they support semi-structured data, querying data with different keys within a document is supported. Entire data in document stores is "queriable" and "indexable". They are eventually consistent for queries by intermittently refreshing the indexes or reading from secondary index while read, writes are performed on primary indexes.

C. Transactions: Most document stores are always atomic and support transactions on single document level.

D. Schema: Document stores do not require a rigid schema. Instead, they are schema agnostic and enforce schema whenever needed

E. *Scaling:* They scale up by sharding collection of documents on different nodes and replicating them. Indexes are divided across shards to achieve better query performance.

Therefore, for models with high performance, flexibility, scalability, schema agnostic and semi-structured data, we choose document stores as they support CP of CAP theorem. MongoDB and Couchbase and popular NoSQL databases using Document data stores. In real-word applications, they are useful for content management systems, e-commerce applications, web or real-time analytics and blogging platforms.

Wide column Store Databases:

Wide column stores, also known as “column families” or “columnar databases”, are designed to store huge numbers of columns. In these databases, data is stored based on column values. As column data can be distributed among clusters easily, they support high scalability and are suitable for data mining and analytics-based applications. Performance can be improved by employing MapReduce framework to process large data distributed on several clusters. They are optimized for queries over large datasets,

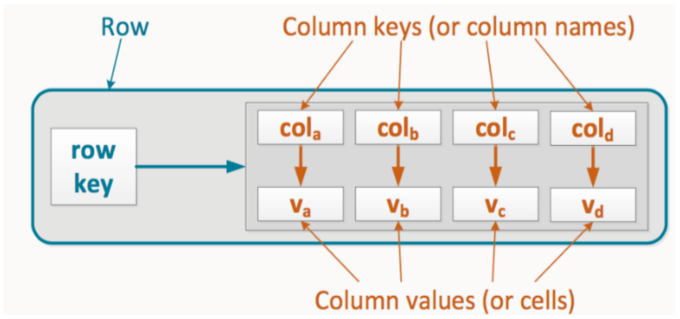


Figure 7 Wide column stores [20]

Evaluation of DB parameters:

A. *Compression:* For data compression and partitioning, Column stores are very efficient.
B. *Query*: Columnar Databases perform greatly with aggregation queries such as COUNT, SUM, and AVG due to their columnar structure.
C. *Scalability*: They are very scalable and more suited for massively parallel processing involving data spread across a large number of machines – often in thousands of clusters.
D. *Fast to load and query:* Columnar stores load the data extremely fast. For example, a billion rows table can be loaded within just few seconds.

Therefore, for models with high performance, flexibility and scalability, we choose document stores as they support CP of CAP theorem. Google’s BigTable, HBase and Cassandra and popular NoSQL databases using Document data stores. In real-world applications, they are useful for content management systems and systems with that need aggregations where heavy write volumes are involved.

Graph Databases:

Graph Databases stores entities or nodes along with their connection or relationships. Occurrence of an object in application is termed as node and each node has a property associated to it. Edges are the lines connecting the nodes to show relations and can have properties. Edges show directional significance and based on these relationship nodes are organized. In short, a node represents an Object whereas edges signify the connection between the objects. Nodes are identified by unique identifier expressing key-value pairs. The core rules of Graph databases are they should always have a start node and an end node. Nodes cannot be deleted without deleting their relationships. Though Graph databases have only single instance of storage, they allow interpretation of data in different ways using the relationships. Moreover, they allow data transformation from one model to other with ease. A Graph Database stores facts along with their relationships, thus making analysis more intuitive.

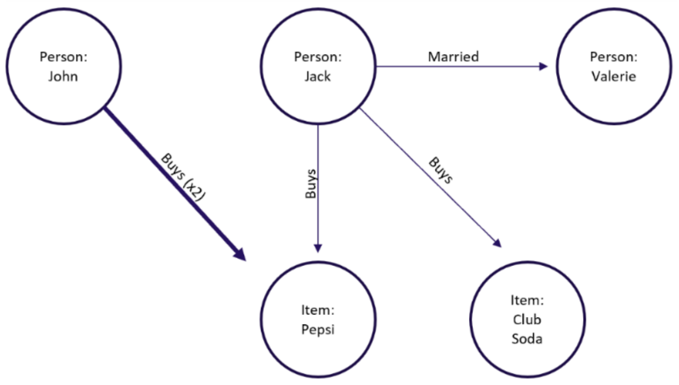


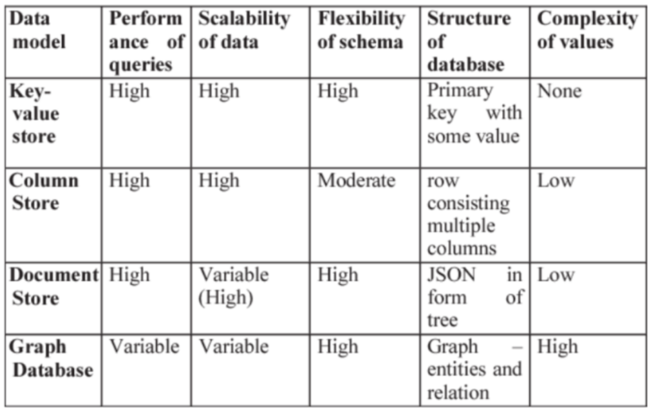
Figure 8 Graph databases store [17]

Therefore, for models which require eventual consistency, flexibility and scalability, we choose document stores as they support CA of CAP theorem. Neo4J and Cassandra and popular NoSQL databases using Document data stores. Graph databases are commonly used to gain insights from relationships between data points or in social media networks, where information to users is determined by their connections. They are also used for building spatial data and recommendation engines.

Evaluation of DB parameters:

A. Query Performance: With growing data where connections grow exponentially, performance of queries in traditional databases is decreased whereas for Graph Databases performance stays constant.
B. Flexibility: With changing requirements, RDMS models forces to change in solutions to tabular way whereas for Graph schemas offer high flexibility.
C. Agility: Developing with aligns with today’s agile methodology i.e. applications having graph-based

backend can evolve with changing business requirements.



Data model	Performance of queries	Scalability of data	Flexibility of schema	Structure of database	Complexity of values
Key-value store	High	High	High	Primary key with some value	None
Column Store	High	High	Moderate	row consisting multiple columns	Low
Document Store	High	Variable (High)	High	JSON in form of tree	Low
Graph Database	Variable	Variable	High	Graph – entities and relation	High

Table 1 Comparison of NoSQL database features [20]

V. Features in Databases RDBMS/NoSQL

Availability: Due to the emerging social media sites such Facebook, Instagram, LinkedIn etc. number of users increased enormously and now handling that huge data is a great challenge. Relational databases suffer from single point failure whereas NoSQL databases are best fit because of their distributed nature when comes to availability.

Consistency: Relational databases are good at providing consistency [22]. To get the uniform view of data after performing operations good consistency is required. To provide consistency, relational databases must sacrifice availability feature.

Cost: NoSQL is an open source which makes it affordable as we don't have to pay for some additional features or upgrades. Relational databases are expensive as they are not open source and we have to pay for every additional feature.

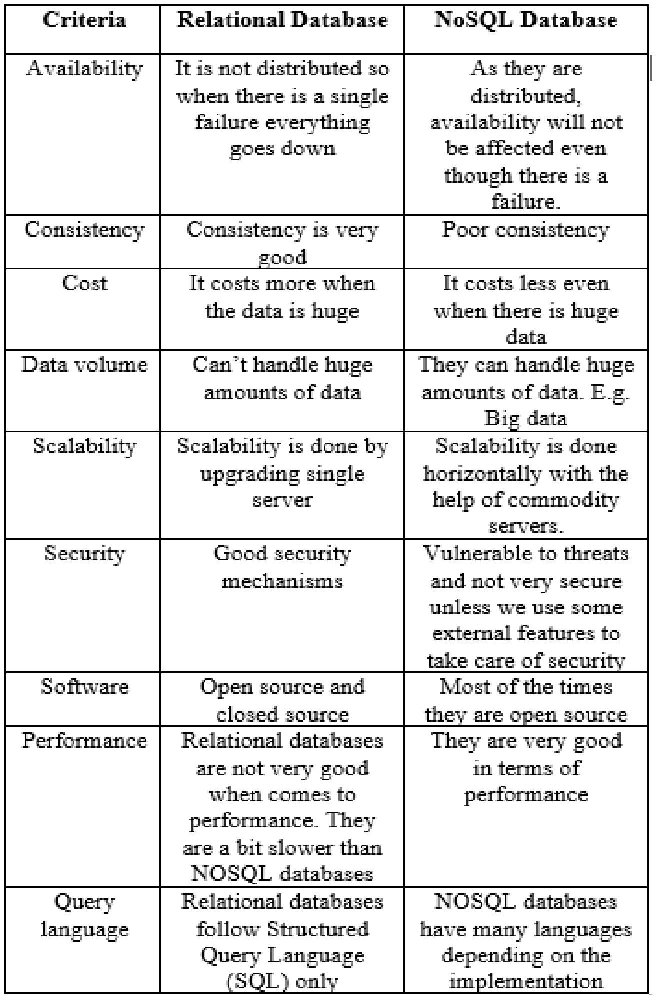
Data Volume: Reputed companies such Google, Facebook are migrating from relational databases to NoSQL databases as it became difficult for them to handle such huge volume of data with relational databases. NoSQL is used when large amounts of data is to be handled.
Scalability: Relational databases are scalable but there is certain limit for the degree of scalability and that limit is predefined by the hardware manufacturer [22]. NoSQL systems are horizontally scalable which is easy and inexpensive. In NoSQL systems no upgrades are required so no additional costs.

Security: Relational databases provide inbuilt authentication and some other security related features. They are good at handling security issues. However, they suffer some challenges related to security such as SQL injection and cross site scripting. NoSQL systems are vulnerable to attacks and those in NoSQL systems security should be handled by middleware [22].

Software: NoSQL is an open source which makes it affordable and providing opportunity for researchers to explore the source code e.g. MongoDB, Cassandra. In relational databases, some are open source, and some are closed platforms.

Performance: NoSQL systems are good at performance as they retrieve data from volatile memory whereas relational databases retrieve data from non-volatile memory which results in slow processing. MongoDB is better than Oracle for most of the database operations.

VI. Comparison of RDBMS and NoSQL



Criteria	Relational Database	NoSQL Database
Availability	It is not distributed so when there is a single failure everything goes down	As they are distributed, availability will not be affected even though there is a failure.
Consistency	Consistency is very good	Poor consistency
Cost	It costs more when the data is huge	It costs less even when there is huge data
Data volume	Can't handle huge amounts of data	They can handle huge amounts of data. E.g. Big data
Scalability	Scalability is done by upgrading single server	Scalability is done horizontally with the help of commodity servers.
Security	Good security mechanisms	Vulnerable to threats and not very secure unless we use some external features to take care of security
Software	Open source and closed source	Most of the times they are open source
Performance	Relational databases are not very good when comes to performance. They are a bit slower than NOSQL databases	They are very good in terms of performance
Query language	Relational databases follow Structured Query Language (SQL) only	NOSQL databases have many languages depending on the implementation

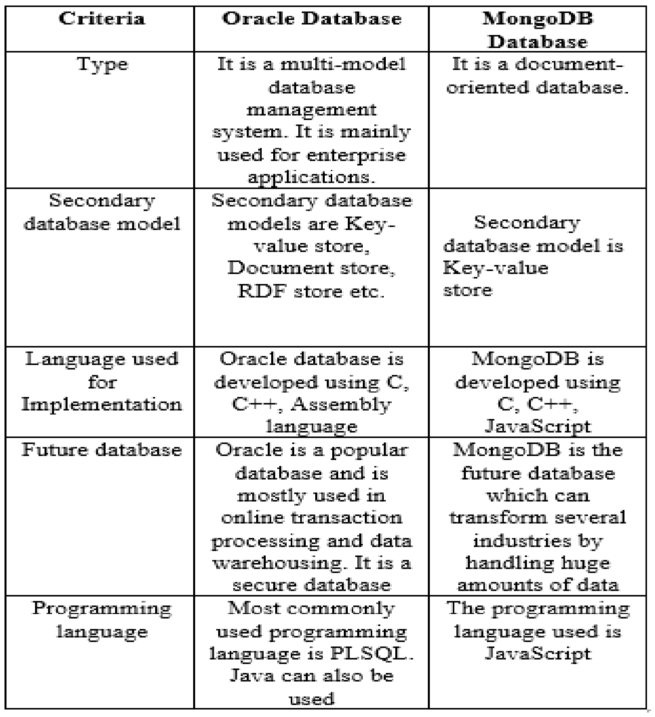
Table 2 Comparison of databases [22]

In the above table comparison [22] is done between relational and NoSQL databases taking different factors into consideration. To be more specific let us consider Oracle from

Relational databases and MongoDB from NoSQL databases for comparison.

Oracle:
Oracle was released by Oracle Corporation as a relational model initially and later extended to object-relational model to store complex business models. Oracle is one of the market leaders in the sector of database. The latest Oracle database version is 12C. C stands for Cloud [24]. Oracle database have some unique features like database consolidation, performance tuning, query optimization, high availability, partitioning, recovery options etc. Consolidation of databases is helpful to optimize hardware and operational staff [24]. It can plug into existing database without making changes in the application's code. There are recent updates to the database to improve data integration with cloud and big data analytics [24].

MongoDB:
MongoDB was developed by MongoDB Inc and a MongoDB server has more than one database. MongoDB is one of the top ranked databases and it is the most popular among NoSQL databases. This is a document-oriented database. It supports JSON format which is very popular, and it supports self-start process because it doesn't need a database administrator to bootstrap [24]. This database offers multi-version concurrency control. This feature allows to keep previous versions of data available to achieve consistency in case of complex transactions. In the below table comparison [23] is done between Oracle and MongoDB databases.



Criteria	Oracle Database	MongoDB Database
Type	It is a multi-model database management system. It is mainly used for enterprise applications.	It is a document-oriented database.
Secondary database model	Secondary database models are Key-value store, Document store, RDF store etc.	Secondary database model is Key-value store
Language used for Implementation	Oracle database is developed using C, C++, Assembly language	MongoDB is developed using C, C++, JavaScript
Future database	Oracle is a popular database and is mostly used in online transaction processing and data warehousing. It is a secure database	MongoDB is the future database which can transform several industries by handling huge amounts of data
Programming language	Most commonly used programming language is PLSQL. Java can also be used	The programming language used is JavaScript

Table 3 Oracle DB vs MongoDB [23]

Query processing: One of the main purposes of storing data in databases is to access/read or write huge amounts of data as fast as possible without any data loss. So, we expect our database to return results for our query and modify the data very quickly and easily.

RDBMS: Relational databases rely on rows, columns, tables and schemas to organize data. Data retrieval also depends on the same. Retrieving data is more complicated and takes lot of time in relational databases.

NoSQL: In NoSQL data is unstructured and they use data models. Data retrieval is fast in this database. Some examples of NoSQL data are user and session data, images, videos, chat data and log data.

VII. Essential Security Features in Databases:

Security plays an important role in the modern database management systems. Database servers are very important in any organization. To ensure everything goes well DBMS has certain features [10].

Identification and authorization: Databases will be accessed by various persons from various sources like scanners, wireless access, internal network, remote access etc. Every user should be authenticated before accessing the databases. This authentication can be done by operating system or from database systems. For example, in MSSQL user can log in with OS credentials or through separate database account but in Oracle database users log in through database account.

Reviewing users and passwords: All database user accounts should be reviewed occasionally, and the purpose of every account should be identified. The role of the user accounts should also be reviewed to make sure no user is having inappropriate privileges. There should be some guidelines for choosing the passwords and users should be restricted to choose simple and vulnerable passwords.

Application Systems Connections: Databases can be connected to the other applications such as oracle apps, JDE, PeopleSoft etc. In most of the cases those connections are hard coded which makes them vulnerable. To handle such cases, we must make sure those hardcoded files are encrypted, or those files can be stored in a user inaccessible file.

Logging and Monitoring: The database usage should be logged and monitored continuously to make sure everything is going smooth and acceptable.

Backup and Recovery: There should be an appropriate backup and recovery strategy. Some of them are partial or full database backup, archive log files, transaction logs etc.

Vulnerability analysis: Vulnerable assessments should be done periodically, and appropriate action should be taken if something doesn't look good.
Encryption: Before storing sensitive information in database, that information should be encrypted, and key will be used to decrypt that information later.
In the table below [12] comparison is done between relational and NoSQL databases with respect to security features.

Security Features Comparison:

Category	**Relational databases**	**NoSQL databases**
Authentication	Most of the relational databases has authentication or authorization mechanism by default.	Most of the NOSQL databases doesn't have authentication or authorization mechanism by default. However, we can achieve this by adding some external features.
Auditing	Some relational databases such as Oracle provides in built audit mechanisms which will enable writing records in log file in some exceptional cases.	Some NOSQL databases provide audit facility but there is chance of security breach.
Data Integrity	Relational databases follow ACID properties. This property always provides data integrity.	NOSQL databases follow CAP theorem. According to CAP theorem only two properties holds out of three consistency, availability and partition tolerance. So, there is no guarantee of achieving data integrity all the time.
Confidentiality	In relational databases we have encryption techniques. From these we can achieve confidentiality in relational databases.	Confidentiality is not as good as relational databases
Client communication	Client communication can be done safely with encryption and SSL protocols.	Client communication is not done efficiently in NOSQL databases

Table 4 Security Features of Relational vs NoSQL [12]

Challenges in securing Oracle:

Patching: Installation of patches is difficult in oracle [13]. Many DBA's faced database crash issues after they install patches.
Web applications: Handling with web applications is difficult as most of the non-commercial web applications are built on open source making them more vulnerable to threats.

Security Issues in MongoDB:

There are certain drawbacks concerned with security in MongoDB design.

1. Data files in MongoDB are not encrypted. MongoDB doesn't have any method to encrypt those files automatically [11]. The files are vulnerable as an attacker having access to the file system can misuse them.
2. A binary wire level protocol using TCP port 27017 is used by default in MongoDB client interfaces [11]. This protocol is not encrypted or compressed.
3. MongoDB mainly uses JavaScript as internal scripting language. There is a possibility of injection attacks as JavaScript is an interpreted language [11].
4. When MongoDB is operating in sharded mode, authentication is not supported.

VIII. Indexing Techniques in NoSQL

Indexing is the process to associate a key for location of a corresponding data record. Indexing structures such as T-Tree, B+ Tree and O2-Tree exist in NoSQL and provides sequential access to main memory which is cheaper than in disk-based systems.

T-Tree:

The T-Tree was proposed by Lehman and Carey in 1986, as a new data structure for main memory database systems. It is developed by combining the features of B-Tree and AVL Tree. B-Trees are unbalanced trees where different number of children are allowed for each node whereas AVL-Trees are self-balancing binary trees. The T-Tree structure is like B-Tree and AVL-Tree.

T-Trees have three types of nodes: Leaf nodes are T-nodes with no children, an internal node are T-nodes with right and left child and half-leaf nodes are T-nodes with one child. Each node contains more than one pair of {Key-value, pointer} tuples. They maintain the binary search tree property of AVL-Tree but has better storage utilization as each node contains several pairs of nodes in sorted order which also produces better performance. Each T-node have two child record points to lower level nodes. While there is similarity in query operations with that of AVL-Tree, T-Trees offer better performance over them.

O2-Tree:

O2 Trees are binary search trees with leaf nodes containing {Key vale, pointers} tuples. O2-Tree of order m, where m is minimum degree of the tree (and greater than or equal to 2), should satisfy these properties [16]:

- A root is black whereas every node is either red or black.
- A red node implies that both its children are black.
- Every leaf node consists of a block which holds "Key value, record-pointer" tuples and is colored in black.
- For each internal node, there are same number of black nodes for all simple paths from node to their descendant leaf nodes.

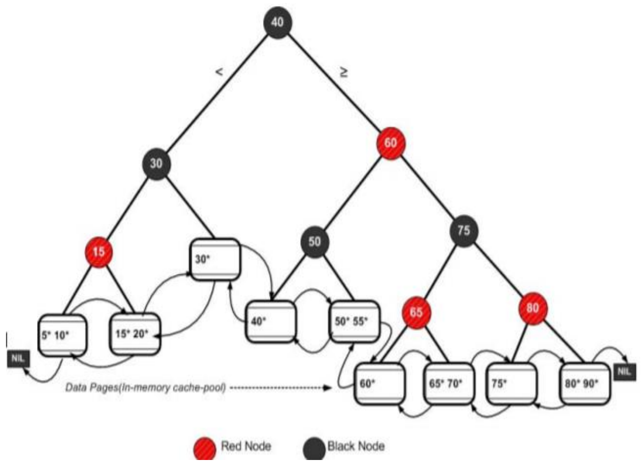


Figure 9 Structure of O2-Tree[16]

- Leaf-nodes have between m/2 and m blocks of {Key-value, pointer} tuples.
- If tree has only single node, it can have data items between 1 to m, and must be a leaf, which is root of tree.
- Leaf nodes are double linked in backward as well as forward directions.

The O2 Tree logarithm time for query processing operations. Even though the tree height is as low as B+ Tree, they perform less comparisons at each internal node as it has lower branching factor (2 per node) which results in faster processing time when compared with indexes such as T-Tree, B+ - Tree, AVL-Tree. Hence O2 Trees are preferred for large data sets requiring main memory indexing structures [16].

IX. Case Study

The following work provides comparison between SQL database management system - Oracle Database and No SQL document oriented database management system - Mongo DB as seen from a user point of view, which includes from system requirements, the operating systems you can install them on, syntax differences, drivers to performance differences in query time, insert time and update time on an identical test database [4].

The primary difference in order to achieve performance or reliability features that are incompatible with their flexibility of SQL, NoSQL is a category of database engines that do not support SQL. Query languages in NoSQL engines provides subset of SQL features and also new features**.** SQL engines provides JOIN, TRANSACTION, LIMIT, WHERE, GROUP BY and ORDER BY, these are not usually supported in NoSQL engines.
Some NoSQL engines support automatic import-from-SQL features, but NoSQL imposes some architecture constraints depending on how we are dealing with the data. It is uncommon that an entire existing SQL-driven application to be integrated over to NoSQL.

Oracle is a relational database model and MongoDB is a NoSQL document-oriented schema-less database model. Oracle database use SQL as query language in comparison with the one used by MongoDB which consists of API calls, JavaScript and REST.

Both MongoDB and Oracle Database use composite keys, conditional entry updates, full text search and Unicode characters. MongoDB has built-in map-reduce function which can be used to aggregate large amounts of data.

MongoDB agrees to larger information of data, the maximum value size in MongoDB is 16 MB when in comparison with Oracle Database which maximum value size is 4 KB. The integrity model used by MongoDB is BASE, while Oracle Database used ACID. Oracle Database provides reliability features like isolation, transactions, referential integrity and revision control, which MongoDB doesn't provide.

MongoDB provides durability, consistency as well as conditional atomicity. Both MongoDB and Oracle Database are horizontal scalable in the way they are distributed and supports for data replication. Oracle database doesn't offer any sharing support whereas MongoDB does.

Oracle database is a licensed product which was written in C++, C and Java, while MongoDB is a free product was written in C++. Both Oracle and MongoDB engines are multiuser, active databases and the databases doesn't have compression support.

Time Comparison:

There are many syntax differences between the two database management systems. Oracle database uses SQL SELECT, UPDATE, and DELETE for data manipulation and MongoDB uses Functions for adding new records, updating and deleting the existing records.

Stored procedures in Oracle databases are used to manipulate the data retrieved from several tables. It uses PL/SQL (Procedural language/Structured Query language) for building advanced queries**.** On the other hand, MongoDB uses callback functions for this kind of advanced queries. JavaScript language is used for declaring functions in mongo shell and these functions can be used to manipulate data returned by the basic queries. Results of these queries can be iterated using cursors and different functions [3].

To compare times between Oracle and MongoDB database engines, we should compare the time it took to perform same actions in both database engines.

The first action is Inserting a new record in the database.
To insert the data in Oracle and MongoDB we should create an object for handling this data. In Oracle this object we call it as Table and in MongoDB it is called Collection
When we are trying to insert a new record in Oracle Database we have to be careful on not to violate any of the constraints because Oracle Databases have multiple types of constraints on tables and in MongoDB there are no constraints available to be defined on the data collection which is more flexible. This is also one of the major differences between a SQL database and a NoSQL database.

The SQL had fixed structure by storing the data in the form of tables and we can even define relations between the tables within the same database (foreign key relations). PRIMARY KEY Constraint can be set on a table and composed of one or more columns from the table. There are other constraints can be declared on the columns of a table, like UNIQUE, FOREIGN KEY or NOT NULL.

NoSQL doesn't have any fixed structure and stores the data using collections. These collections have no constraints regarding the data stored in them (with in the collection fields can have different data types).

In Oracle database table is created using the below CREATE statement:

```
CREATE TABLE users (
 user_id int not null,
 first_name varchar2(50),
 last_name varchar2(50)
);
```

Inserting a new record in OracleDB looks like below:

```
INSERT INTO users (user_id, first_name, last_name)
VALUES (1, 'Newfirstname1','Newlastname1')
```

And in MongoDB collection is created in the first insert, uses functions and BSON objects for inserting a new record. The functions used for inserting new record in existing collection are 'insert' and 'save'. A new field is automatically added to new object while inserting a new object in MongoDB and this field is called '_id' and is unique (can be used like an index).

Below is the mongo shell script for adding new record to the 'user' collection:

```
db.users.insert({
    user_id: 1,
    first_name: 'Newfirstname1'
    last_name: 'Newlastname1'
});
```

We can observe from above structure doesn't have to be fixed because of the flexibility of collection.

Below results (time in milliseconds, for each set of inserts) are obtained from generated records that are inserted in the database, at first took a set of 10 records and computed across the two databases on how much time it took to insert the rows in database. After that compared the time for set of 100 records, 1000, 10000, until we got to 1000000.

No. of records	Oracle Database	MongoDB
10	31	800
100	47	4
1000	1563	40
10000	8750	681
100000	83287	4350
1000000	882078	57871

Table 5 for Insertion [2]

From the above Table, we can say that MongoDB is efficient when inserting a large amounts of data but took extra time in inserting first 10 records.
On the other hand, Oracle Database is efficient with little amount of records to be inserted and when we see at time it took for inserting greater than 10000 records, the time spent is very high.

Below figure shows the time spent on insert operation for Oracle and MongoDB databases:

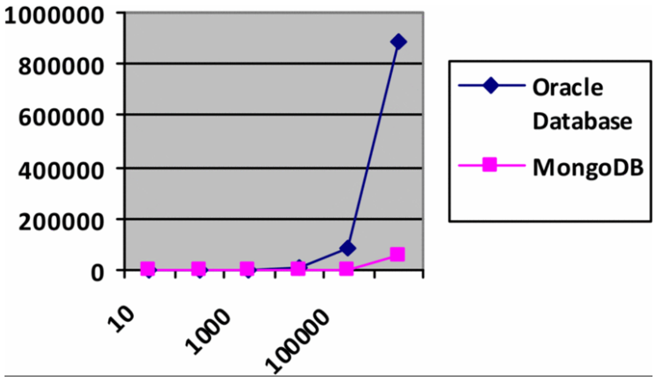


Figure 10 Insertion times [2]

In MongoDB we have 'find' function for retrieving the data. To retrieve all the items from the collection 'users' we use below syntax:

```
 db.users.find();
```

For refining the results we have to specify extra object like:

```
 db.users.find({first_name: "Stevens"});
```

To retrieve only the required fields in MongoDB we can add an object to specify in displaying them. The "_id" field in MongoDB automatically increments the new objects inserted into the collections, you have to specify not to display it if we don't want this information and for the other fields you have to specify to display them [2].

For example, if we want to retrieve data only the "last name" and "first name" and not the "user_id" and the"_id", we use below statement:

```
 db.users.find( {}, {"_id":0, "last_name": 1, "first_name":
1});
```

We can sort the data in MongoDB using sort function where we need to specify the field that we want to the sort the data by and the sorting order: increasing (1) or decreasing (-1):

```
 db.users.find().sort({"last_name":1});
```

The second operation we would like to compare times between both databases is by updating some records, like in the insert operation we do the same for update and check the time it took across both the database engines. Update operation results are carried on the same Inserted records used above.

Update operation in OracleDB looks like below:

```
 UPDATE users
 SET last_name = 'Newlastname1'
 WHERE user_id > 10
 AND user_id <=110;
```

And in MongoDB, the function 'update' helps in updating the existing content in collection. Below statement used to update the existing records on MongoDB:

```
  Db.users.update (
 {user_id: {'$gt': 10, '$lte': 110}},
 {'$set': {last_name: 'Newlastname1'}});
```

We compared the time took for updating the first 10 records (updated the records by giving the condition 'user_id' less than 10) and for the next 100 records (use the condition 'user id' between 11 and 110), for next 1000 updates the 'user_id' between 111 and 1110 and likewise we checked for 10000, 100000 and the field updated is 'last_name'

We see that from the below table, update operation in MongoDB took nearly same for 10, 100 or 1000000 records and on Oracle Database the time got increased with the increase of number of records to be updated.

No. of records	Oracle Database	MongoDB
10	453	1
100	47	1
1000	47	1
10000	94	1
100000	1343	2
1000000	27782	3

Table 6 for Updation [2]

Below figure shows the time spent on Update operation for Oracle and MongoDB databases:

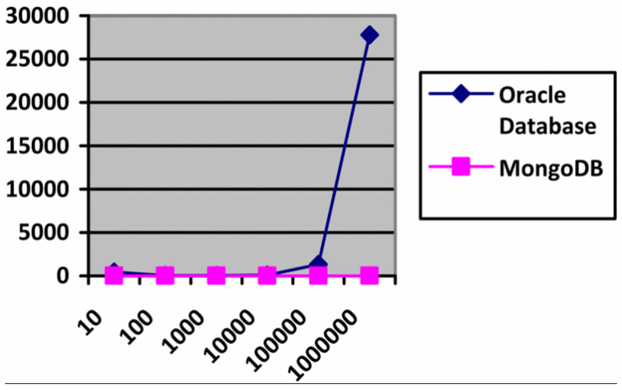


Figure 11 Update times [2]

The last operation we would like to compare the times between both databases is by removing some records. Below results are obtained after removing 10 records, then 100, 1000, 10000, 100000 and 1000000.

Delete operation in Oracle DB looks like below:

```
DELETE from users
WHERE user_id > 10 and user_id <=100;
```

Data entries in MongoDB collection can be deleted using 'remove' function:

```
db.users.remove (
        {user_id: {'$gt': 10, '$lte': 110}});
```

Below table shows the relative times between both the databases:

No. of records	Oracle Database	MongoDB
10	94	1
100	47	1
1000	62	1
10000	94	1
100000	1234	1
1000000	38079	1

Table 7 for Deletion [2]

Below figure shows the time spent on Deleting records on Oracle and MongoDB databases:

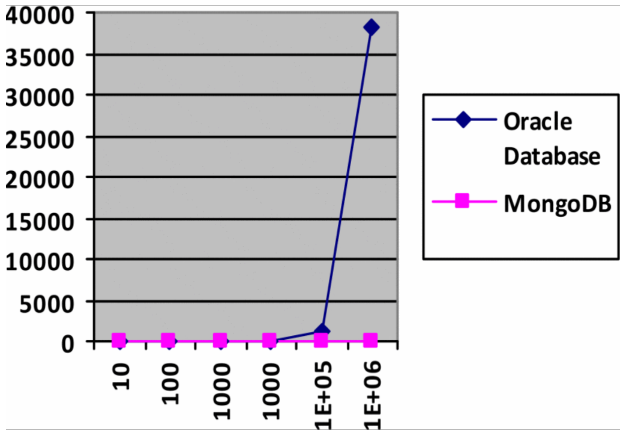


Figure 12 Deletion times [2]

X. Tradeoffs

Scenarios in which NoSQL/MongoDB is preferred:

- Companies are moving towards NoSQL databases because they can build faster applications and maintain applications more efficiently. NoSQL systems are budget friendly.
- There is a rise in the use of unstructured data and predictors say the unstructured data will keep on increasing in the future [24]. Schema is predefined in relational databases whereas NoSQL databases have dynamic schema which makes them easier to handle unstructured data. So, we must choose NoSQL database when we have unstructured data.
- Gigantic companies like Cisco, RBS, and Telefonica, China Eastern moved from Oracle to MongoDB. Developer productivity can be increased by using **MongoDB Stitch** and **MongoDB Mobile**. MongoDB Stitch is a server less platform where backend jobs will be taken care by reducing the development time to half [21]. Stitch Query is used to execute any MongoDB query and Stitch Tigger is used for more flexibility and it is easy to maintain. Oracle doesn't have any of these features and lot of time is wasted there to achieve the above functionality. With MongoDB Mobile, the apps run faster, and they keep on running even if the network is lost. Oracle doesn't have mobile Oracle Enterprise database [21].
- For social media sites NoSQL is recommended over relational databases because for those sites flexibility is more important than consistency [22].
- It is wise to choose NoSQL when dealing with huge amounts of data.

Scenarios in which RDBMS/Oracle is preferred:

- Relational databases are vertically scalable whereas NoSQL databases are horizontally scalable [24]. NoSQL queries focus on collection of documents and Relational queries use SQL to manipulate data. Relational databases are best fit for complex queries when compared to NoSQL.
- When we are looking for a secure client communication and confidentiality then we must go for relational databases.
- Data integrity is achieved in RDBMS.
- Authentication is provided by default is relational databases.

XI. Conclusion

A right database system is chosen based on the set of desirable properties over the remaining systems. The binary decision tree helps to break the complexity of this choice by mapping the trade-off decisions to potentially suitable database systems and applications. In the first split of the tree, applications either rely of fast lookups or require complex query capabilities. Fast loop applications can further be divided by volume of data they process. If main memory of single machine can hold all the data volume, single node systems like Redis is best choice if functionality is favored or Memcache if simplicity is needed. However, if data volume is unbounded or grow beyond RAM capacity, a multi-node system with horizontal scaling is more appropriate.

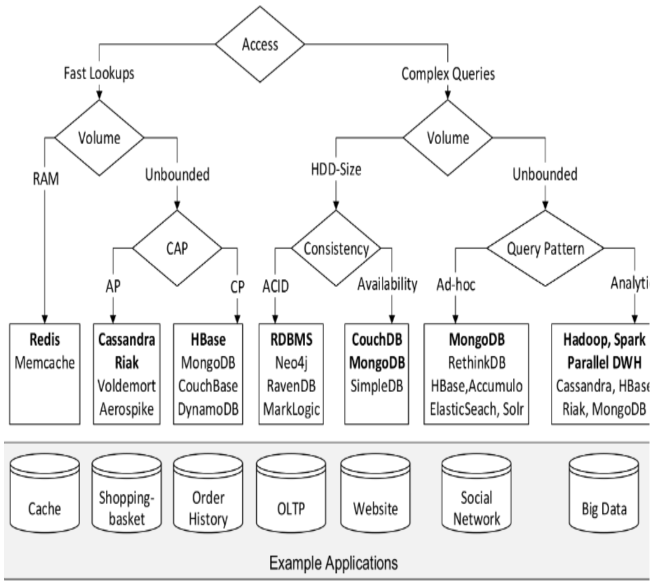


Figure 13 Decision tree for choosing NoSQL database systems based on requirement [19].

To summarize, the right half of the decision tree splits applications that require complex queries, after distinguishing the systems by data volume size and analyze if single single-node systems are feasible or if distribution systems are needed due to unbounded volume capacity. OLTP systems such as online transaction processing-based applications have workloads on large data volumes and hence required ACID semantics. Relational Databases or Graph Databases like Neo4J are optimal for their workloads. However, if availability is priority, distribution system like CouchDB or MongoDB are preferable.

For larger data volume systems, choosing the right database depends on query patterns i.e. when complex queries are to be optimized based on latency [19], MongoDB is preferable as it facilitates ad-hoc queries for social networking applications. However, for Big Data analytics which are throughput optimized, HBase and Cassandra are useful as they excel when combined with Hadoop. As described by CAP theorem, the most important case in choosing the right database system is based on favoring the Consistency (CP) or availability (AP). Riak and Cassandra delivers always-on experience whereas HBase, DynamoDB and MongoDB delivers strong consistency.

The findings revealed that, RDBMS is based on ACID model which signifies for better consistency, security and offers a standard query language. Nevertheless, Relational Databases have weak performance, poor scalability, more cost, and lots of challenges in supporting large number of users and handle minimal volumes of data. On the other hand, NoSQL is based on BASE model, which emphasizes on better performance, greater scalability and schema flexible but, lacks a standard procedural/query language and does not provide adequate security mechanisms. The entire choice of database will depend on the usage of application being developed and both the databases will continue to exist alongside each other with none being better when compared to other.

XII. References

1. Yishan Li, Sathiamoorthy Manoharan, "A performance comparison of SQL and NoSQL databases", Communications Computers and Signal Processing (PACRIM) 2013 IEEE Pacific Rim Conference.
2. MongoDB vs Oracle - database comparison https://ieeexplore.ieee.org/document/6354766/citations#citations
3. A comparative study of NoSQL and Relational Database https://www.researchgate.net/publication/326019759_A_Comparative_Study_of_NoSQL_and_Relational_Database
4. SQL vs NoSQL: Which one is better to use? https://www.geeksforgeeks.org/sql-vs-nosql-which-one-is-better-to-use/
5. Introduction to RDBMS

http://www.rjspm.com/PDF/BCA-428%20Oracle.pdf

6. MongoDB architecture https://searchdatamanagement.techtarget.com/definition/MongoDB
7. Data Migration Between Different Data Models Of Nosql Databases https://acervodigital.ufpr.br/bitstream/handle/1884/49087/R%20-%20D%20 %20LEANDRO%20DUARTE%20PULGATTI.pdf?sequence=1&isAllowed=y
8. NoSQL – ACID Properties and RDBMS Story https://cloudxlab.com/assessment/displayslide/342/nosql-acid-properties-and-rdbms-story
9. Choosing the right NoSQL database for the job: a quality attribute evaluation https://www.researchgate.net/publication/282519669_Choosing_the_right_NoSQL_database_for_the_job_a_quality_attribute_evaluation
10. Security and Control Issues within Relational Databases David C. Ogbolumani, CISA, CISSP, CIA, CISM Practice Manager – Information Security http://www.cpd.iit.edu/netsecure08/DAVID_OGBOLUMANI.pdf
11. Security Issues in NoSQL Databases https://www.researchgate.net/publication/254018091_Security_Issues_in_NoSQL_Databases/link/548977500cf2ef3447926025/download
12. Relational vs. NoSQL Databases: A Survey Mohamed A. Mohamed, Obay G. Altrafi, Mohammed O. Ismail https://www.researchgate.net/publication/263272704_R`3a495312ad22000000/download
13. The Top Five Challenges In Securing Oracle Databases https://www.darkreading.com/risk/the-top-five-challenges-in-securing-oracle-databases/d/d-id/1134824
14. SQL server delayed durability ACID properties https://sqlserver-help.com/2014/05/01/sql-2014-learning-series-7new-feature-delayed-durability-part-1/
15. NoSQL Databases: a Survey and Decision Guidance https://medium.baqend.com/nosql-databases-a-survey-and-decision-guidance-ea7823a822d
16. O2-Tree: A Fast Memory Resident Index for NoSQL Data-Store Daniel Ohene-Kwofie ; E.J. Otoo ; Gideon Nimako | 2012 IEEE 15th International Conference on Computational Science and Engineering
17. Graph Databases. What's the Big Deal https://towardsdatascience.com/graph-databases-whats-the-big-deal-ec310b1bc0ed
18. Exploring different types of NoSQL https://www.3pillarglobal.com/insights/exploring-the-different-types-of-nosql-databases
19. A definitive guide to NoSQL Databases https://www.toptal.com/database/the-definitive-guide-to-nosql-databases
20. NoSQL databases: Critical analysis and comparison Adity Gupta ; Swati Tyagi ; Nupur Panwar ; Shelly Sachdeva ; Upaang Saxena 2017 International Conference on Computing and Communication Technologies for Smart Nation (IC3TSN)
21. MongoDB and Oracle Compared https://www.mongodb.com/compare/mongodb-oracle
22. A Comparative Study of NoSQL and Relational Database Douglas Kunda, Hazael Phiri https://www.researchgate.net/publication/326019759_A_Comparative_Study_of_NoSQL_and_Relational_Database/link/5be95f9b92851c6b27b89bc2/download
23. MongoDB vs Oracle https://www.educba.com/mongodb-vs-oracle/
24. SQL vs NoSQL: From Oracle to MongoDB, which database is right for your business? https://www.cbronline.com/cloud/sql-vs-nosql-from-oracle-to-mondgodb-which-database-is-right-for-your-business-4850321/
25. CAP theorem definition https://medium.com/system-design-blog/cap-theorem-1455ce5fc0a0