NoSQL Database - MongoDB

Dr. K. Amarendra[1], A. Vasudeva Rao [2]
[1] Professor, Dept of CSE
[2]Assoc Prof, Dept of CSE
[1]K L University, Vijayawada, India
[2]Dadi Institute of Engineering & Technology, Visakhapatnam, India

Abstract- *As the data volume is growing tremendously every day, the storage of information, support and maintenance have become difficult. The data stored and updated on daily bases is in the form of logs, audio, video, sensor data and so on, which is not easy to be stored and queried using relational database. This paper gives the overview of NOSQL databases which provide more scalability and efficiency in storage and access of the data. This paper also provides comparative study between SQL and No- SQL.*

Keywords- NoSQL,MongoDB,RDBMS,SQL

I. INTRODUCTION

Relational databases have been around since the 70's so they're a very mature technology. In general they support transactions allowing you to make changes to your data in discrete, controlled manner, they support constraints such as uniqueness, primary and foreign keys, and check constraints[4]. And furthermore they use SQL or so-called Simplified Query Language to access ie fetch data, and also modify data by inserting, updating or deleting records.

Enter NOSQL databases like MongoDB which attempt to address some of these concerns. For starters data is not read/written to the database using the old SQL language, but rather using an object-oriented method which is developers find very convenient and intuitive. What's more it supports a lot of different type of indexing for fast lookups of specific data later.

But NOSQL databases don't just win fans among the development side of the house, but with Operations too, as it scales very well. MongoDB for instance has clustering built-in, and promises an "eventually consistent" model to work against.To be sure a lot of high-profile companies are using NOSQL databases, but in general they are in use for very specific needs.

NoSQL is an approach to databases that represents a shift away from traditional relational database management systems (RDBMS). To define NoSQL, it is helpful to start by describing SQL, which is a query language used by RDBMS. Relational databases rely on tables, columns, rows, or schemas to organize and retrieve data. In contrast, NoSQL databases do not rely on these structures and use more flexible data models. NoSQL can mean "not SQL" or "not only SQL." As RDBMS have increasingly failed to meet the performance, scalability, and flexibility needs that next-generation, data-intensive applications require, NoSQL databases have been adopted by mainstream enterprises[5]. NoSQL is particularly useful for storing unstructured data, which is growing far more rapidly than structured data and does not fit the relational schemas of RDBMS. Common types of unstructured data include: user and session data; chat, messaging, and log data; time series data such as IoT and device data; and large objects such as video and images.

II. FEATURES OF NOSQL DATABASE

Features of NoSQL

It reads and writes data fast. It is simpler to expand and it also supports massive storage with a low price[8].

Data model

It has a key value data model which means there is a value which leads to key, the structure is simple with an increase in query speed.

It has a column oriented database that is a table but it does not support table association that is a join. Each and every column is the index of the database.

There is a database in which there are collections and in collections columns and in columns documents are there. Classification is done on the basis of CAP theorem.

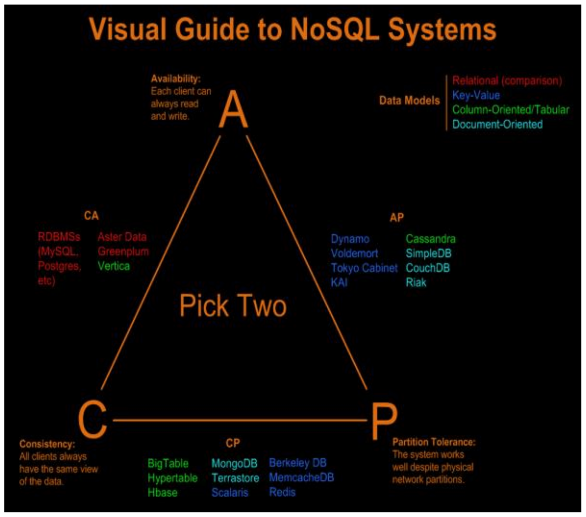
CAP Theorem

It says that there could be three properties of a shared data system that is regularity, scalability and availability, and tolerance that is there in network partitions[6].

You've got a few choices when addressing the issues thrown up by CAP. The obvious ones are:

1. Drop Partition Tolerance

If you want to run without partitions you have to stop them happening. One way to do this is to put everything (related to that transaction) on one machine, or in one atomically-failing unit like a rack. It's not 100% guaranteed because you can still have partial failures, but you're less likely to get partition-like side-effects. There are, of course, significant scaling limits to this.

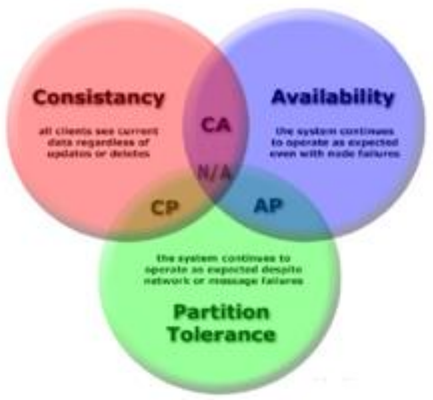


2. Drop Availability

This is the flip side of the drop-partition-tolerance coin. On encountering a partition event, affected services simply wait until data is consistent and therefore remain unavailable during that time. Controlling this could get fairly complex over many nodes, with re-available nodes needing logic to handle coming back online gracefully.

3. Drop Consistency

Vogels' article is well worth a read. Lots of inconsistencies don't actually require as much work as you'd think (meaning continuous consistency is probably not something we need anyway). In a book order example if two orders are received for the one book that's in stock, the second just becomes a back-order. As long as the customer is told of this (and remember this is a rare case) everybody's probably happy.


Consistancy
all clients see current data regardless of updates or deletes
Availability
the system continues to operate as expected even with node failures
CA
N/A
CP
AP
the system continues to operate as expected despite network or message failures
Partition Tolerance

III. TYPES OF NOSQL DATABASES

Several different varieties of NoSQL databases have been created to support specific needs and use cases. These fall into four main categories[9]:

Key-value data stores: Key-value NoSQL databases emphasize simplicity and are very useful in accelerating an application to support high-speed read and write processing of non-transactional data. Stored values can be any type of binary object (text, video, JSON document, etc.) and are accessed via a key[7]. The application has complete control over what is stored in the value, making this the most flexible NoSQL model. Data is partitioned and replicated across a cluster to get scalability and availability[3]. For this reason, key value stores often do not support transactions. However, they are highly effective at scaling applications that deal with high-velocity, non-transactional data.

Document stores: Document databases typically store self-describing JSON, XML, and BSON documents. They are similar to key-value stores, but in this case, a value is a single document that stores all data related to a specific key. Popular fields in the document can be indexed to provide fast retrieval without knowing the key. Each document can have the same or a different structure.

Wide-column stores: Wide-column NoSQL databases store data in tables with rows and columns similar to RDBMS, but names and formats of columns can vary from row to row across the table. Wide-column databases group columns of related data together. A query can retrieve related data in a single operation because only the columns associated with the query are retrieved. In an RDBMS, the data would be in different rows stored in different places on disk, requiring multiple disk operations for retrieval.

Graph stores: A graph database uses graph structures to store, map, and query relationships. They provide index-free adjacency, so that adjacent elements are linked together without using an index.

IV. BENEFITS OF NOSQL

NoSQL databases offer enterprises important advantages over traditional RDBMS, including[8]:

Scalability: NoSQL databases use a horizontal scale-out methodology that makes it easy to add or reduce capacity quickly and non-disruptively with commodity hardware. This eliminates the tremendous cost and complexity of manual sharding that is necessary when attempting to scale RDBMS.

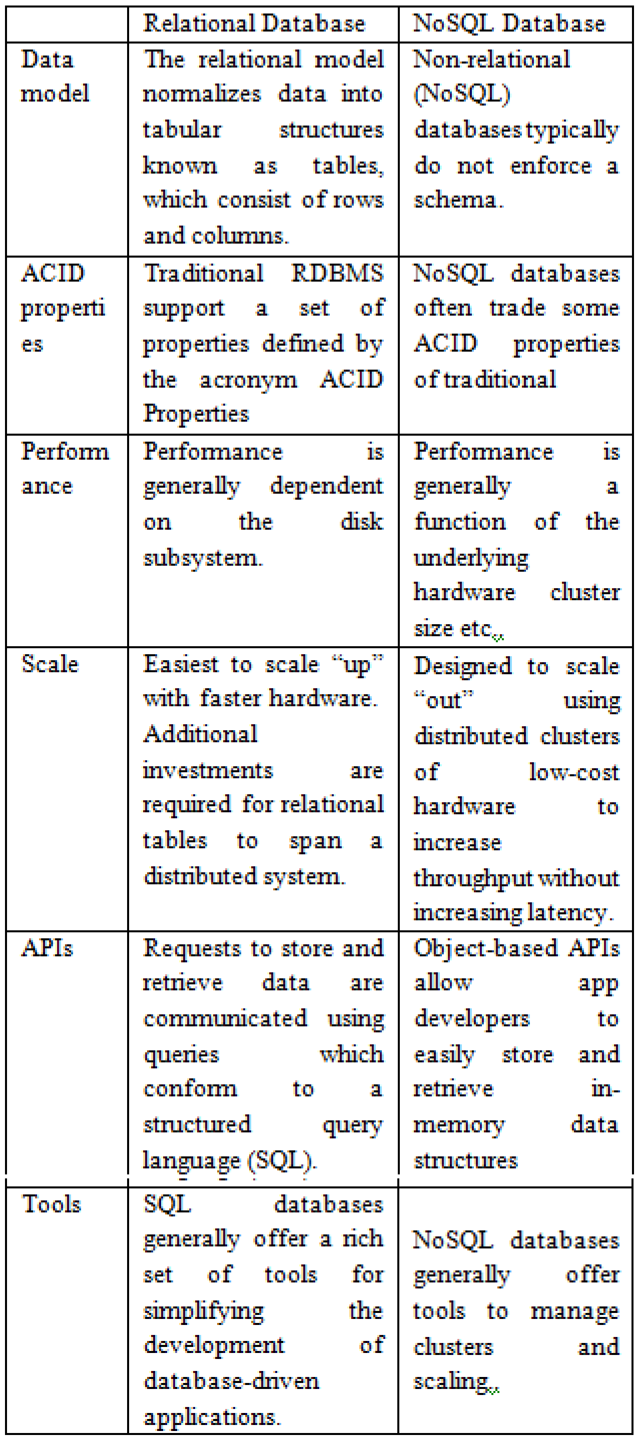
Performance: By simply adding commodity resources, enterprises can increase performance with NoSQL databases. This enables organizations to continue to deliver reliably fast user experiences with a predictable return on investment for adding resources—again, without the overhead associated with manual sharding.

High Availability: NoSQL databases are generally designed to ensure high availability and avoid the complexity that comes with a typical RDBMS architecture that relies on primary and secondary nodes. Some “distributed” NoSQL databases use a masterless architecture that automatically distributes data equally among multiple resources so that the application remains available for both read and write operations even when one node fails.

Global Availability: By automatically replicating data across multiple servers, data centers, or cloud resources, distributed NoSQL databases can minimize latency and ensure a consistent application experience wherever users are located. An added benefit is a significantly reduced database management burden from manual RDBMS configuration, freeing operations teams to focus on other business priorities[5].

Flexible Data Modeling: NoSQL offers the ability to implement flexible and fluid data models[3]. Application developers can leverage the data types and query options that are the most natural fit to the specific application use case rather than those that fit the database schema. The result is a simpler interaction between the application and the database and faster, more agile development.

V. COMPARATIVE STUDY BETWEEN SQL & NO SQL



	Relational Database	NoSQL Database
Data model	The relational model normalizes data into tabular structures known as tables, which consist of rows and columns.	Non-relational (NoSQL) databases typically do not enforce a schema.
ACID properties	Traditional RDBMS support a set of properties defined by the acronym ACID Properties	NoSQL databases often trade some ACID properties of traditional
Performance	Performance is generally dependent on the disk subsystem.	Performance is generally a function of the underlying hardware cluster size etc.
Scale	Easiest to scale “up” with faster hardware. Additional investments are required for relational tables to span a distributed system.	Designed to scale “out” using distributed clusters of low-cost hardware to increase throughput without increasing latency.
APIs	Requests to store and retrieve data are communicated using queries which conform to a structured query language (SQL).	Object-based APIs allow app developers to easily store and retrieve in-memory data structures
Tools	SQL databases generally offer a rich set of tools for simplifying the development of database-driven applications.	NoSQL databases generally offer tools to manage clusters and scaling.

VI. MONGODB

MongoDB is an open source NOSQL database which falls under the classification of document database. It was initiated by 10gen Company. It was written in C++. MongoDB documents are stored in binary form of JSON called BSON format. BSON supports string, integer, date, Boolean, float and binary types. MongoDB is schema less so it gives the freedom to user for inserting new fields to the document or updating the previous structure of document. It does not use joins like relational databases as it has embedded documents which can be accessed quickly. MongoDB also support Master Slave replication where slave nodes contains the replicas of master nodes and are used for backups and reads. MongoDB cluster is built up of three components[1].

Shard node: it contains one of more shards. Each master shard has its one or more slave shards. The slave shards contains the copies of actual data which can be used at the time of failures[1]. The read/ write operation is done by using the data present at the master node. Configuration Server: A group of servers in MongoDB are called the configuration servers. The role of configuration server is to store the routing information and send the request to the right shard. Mongos: Mongos are stateless and can be run in parallel [8]. It is responsible for serving the task requests from the clients. The request can need multiple shards. So mongos collects the data from different shards and merge them nicely before handing it over to the client. Some features of MongoDB are [9]:

Flexibility: MongoDB stores data in document format using JSON. It is a schema less document and maps to native programming language types.

Rich query language: It gives the feature of RDBMS what we are used to with additional features of its own. Dynamic queries, sorting, secondary indexes, rich updates, easy aggregation, upsert (update if document exists and insert if it does not) are few RDBMS features and flexibility and scalability are the additional ones.

*Sharding:*Autosharding allow us to scale our cluster linearly by adding more machines. It is possible to increase the efficiency which is very important on the web when load can increase suddenly and bring down the website[2].

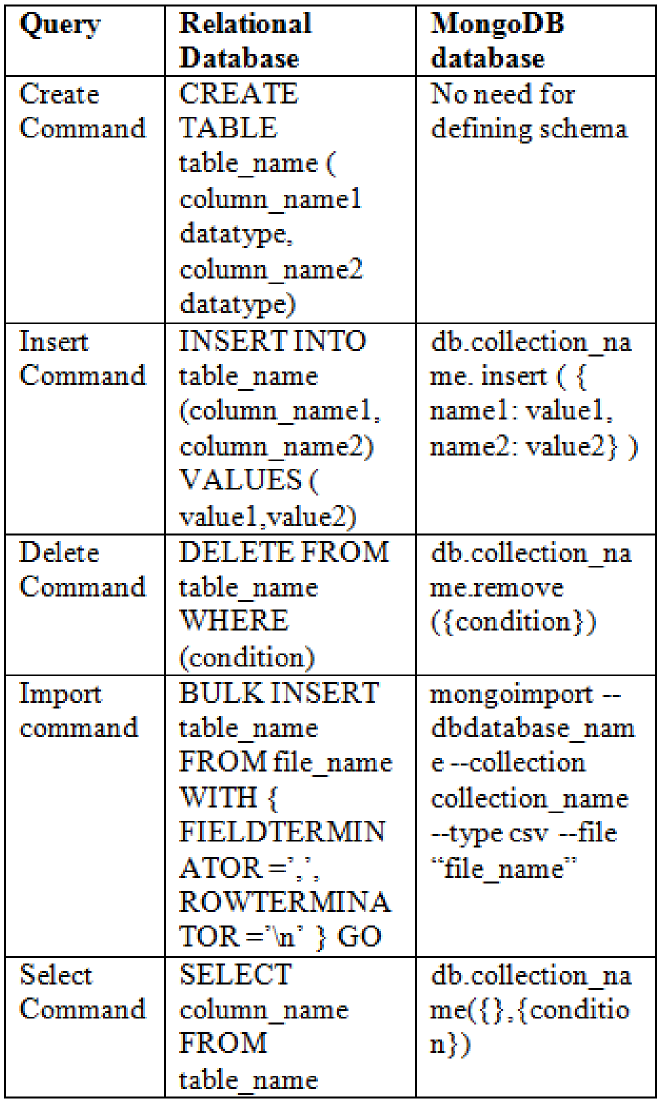
Ease of use: It is very easy to install, use, maintain and configure.

High performance: It provides high performance data persistence. It reduces I/O activity on database system by supporting embedded documents. Use of indexing supports faster queries.

High availability: MongoDB support replication facility called, replica set. Relica set is a group of servers that maintains same dataset. It provides automatic failover, redundancy and increased data availability.

Support for multiple storage engines: It supports multiple storage engines such as WiredTiger storage engine, MMAPv1 storage engine. It also supports pluggable storage engine API that allows third party to develop storage engine for MongoDB[9].

The table Table1 below shows the terms and the concepts of two different types of database relational database such as MySQL and NOSQL database, here MongoDB[2].



Query	**Relational Database**	**MongoDB database**
Create Command	CREATE TABLE table_name (column_name1 datatype, column_name2 datatype)	No need for defining schema
Insert Command	INSERT INTO table_name (column_name1, column_name2) VALUES (value1,value2)	db.collection_name. insert ({ name1: value1, name2: value2})
Delete Command	DELETE FROM table_name WHERE (condition)	db.collection_name.remove ({condition})
Import command	BULK INSERT table_name FROM file_name WITH { FIELDTERMINATOR =',', ROWTERMINATOR ='\n' } GO	mongoimport --dbdatabase_name --collection collection_name --type csv --file "file_name"
Select Command	SELECT column_name FROM table_name	db.collection_name({},{condition})

VII. SECURITY IN MONGODB

In MongoDB the data files are unencrypted in nature and it does not provide an implementation to encrypt the files automatically[1].

It uses a binary wire level protocol which uses TCP port 27017 by default. It is at high verge of injection attacks because it uses mostly java script language. Thus, any attacker can easily acquire the passwords of the data files of user in a particular database.

No auditing is allowed here.

In data communication no encryption is there[2].

The main drawback in both of the NoSQL database was the lack of encryption, weak authentication between the client and the server, denial-of-service and the vulnerability to injection in the database. To decrement the injection in the database it should be verified that the application does reasonable input validation.

The data at rest that is unencrypted can be protected with the OS level mechanism.

VIII. CONCLUSION

NoSQL database are faster and better than SQL database in many ways, ranging from speed to flexibility. Also, an existing database in SQL can be transferred into a MongoDB (NoSQL) with quite an ease and these are the reasons many companies are shifting their projects so as to use MongoDB instead of traditional SQL database. The Erwin HAWK tool can be used to choose a suitable NoSQL database and replace the traditional relational database. NoSQL is also called the future of data economy.

MongoDB is not only faster in most of the queries a Oracle database faces, it is also more flexible and can store large data with ease. It’s flexibility and speed is the main up-votes to switch to MongoDB. Also, as mentioned earlier one can migrate over to MongoDB from SQL based Oracle, with little difficulties.

Taking everything into account, on the off chance that if one needs to utilize a quick, adaptable database, he or she can depend on MongoDB. In the event that the speed of the database isn't a principle concern, and on the off chance that relations are required between the tables and the accumulations, one should depend on the great, Oracle Database.

REFERENCES

[1] MongoDBDocumentation https://docs.mongodb.com./
[2] MongoDB https://www.mongodb.com./
[3] Rupali Arora, Rinkle Rani Aggarwal, “ An Algorithm for Transformation of Data from MySQL to NoSQL (MongoDB)” *IJASCSE*, Volume 2, Special Issue 1, 2013.
[4] Jing Han, Haihong E, Guan Le, Jian Du - Survey on NoSQL Database IEEE2011
[5] AlexandruBoicea, Florin Radulescu, Laura IoanaAgapin - MongoDB vs Oracle - database comparison. IEEE2012
[6] Han, J.: Survey on NOSQL Databases. Proceedings 6th International Conference on PervasiveComputing and Applications, pp. 363-366.
[7] Gajendran, S.: A Survey on NoSQL Databases, 2012, http://ping.sg/story/A-Survey-on-NoSQLDatabases-Department-of-Computer-Science.
[8] Stonebraker, M.: SQL databases vs. NoSQL databases. Communications of the ACM, Vol. 53 No. 4,Pages 10-11.
[9] Rabi Prasad Padhy, Manas Ranjan Patra, Suresh Chandra Satapathy, “RDBMS to NOSQL:Reviewing some next generation Non Realtional databases”, International Journal of Advanced Engineering Science and Technology, Volume 11, Issue 1, 201