**CST2102 Database Analytics**

NoSQL Databases –

An interesting look at the three dominant players

By

Tao YANG, S#040926874

Ying REN, S#041032669

Weixin CEN, S#041027586

Equally contributed.

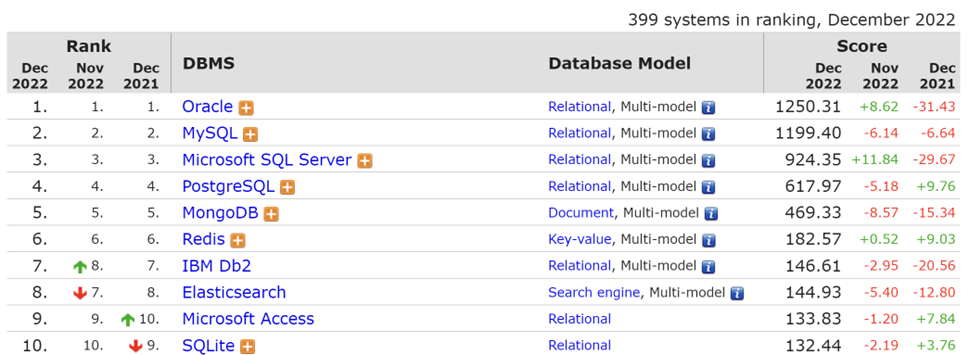
December 06, 2022

## **Introduction to NoSQL**

NoSQL database stands for “not only SQL”, are non-tabular databases which store data differently than relational DBMSs. There are various different types ot NoSQLs, including key-value store, document-oritented, graph database and so on. Compared to the traditional relational database, NoSQL database usually has higher performance, better scalability, higher flexibility and therefore, NoSQL becomes more and more popular in the era of big data. In the following sections, we will introduce three dominant NoSQL products.

**2.** **MongoDB**

Different from the traditional relational DBMS, MongoDB is a document-oriented data model and a non-structured query language. According to the popularity rank in *db-engines.com*, MongoDB is the most popular NoSQL database. MongoDB is an open-source database which supports various popular languages, including C++, C, python, Java and so on.



MongoDB has been designed with developer productivity and flexibility in mind. So, it adopts a document-oriented form, which means the data is stored as documents, and documents are grouped in collections. MongoDB is often referred to as a ‘schemaless‘ database because the documents in a single collection need to have the same set of fields is not a necessary requirement. This feature ensures both productivity and flexibility.

**2.1 Description of Architecture**

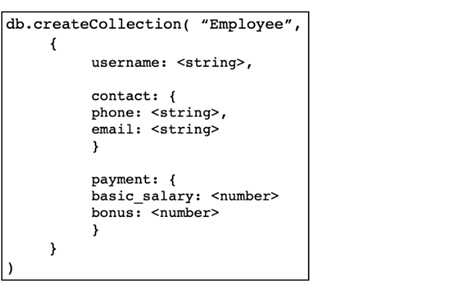
MongoDB is the most SQL-like database among NoSQL, as the comparison shown in the table below. The architecture of MongoDB consists of three levels, database, collection, and documents. A database is a container of data. Each database has its own set of files. Multiple databases can be under a single MongoDB Server. In each database, there are multiple collections which is like tables in RDBMs. A group of database documents is a collection. Inside a collection, different documents can have different fields, but usually mostly documents in the same collection should have same purpose or have the same goal.

| **SQL Concept** | **MongoDB Concept** |
| --- | --- |
| Database | Database |
| Table | Collection |
| Row | Document |
| Column | Field |

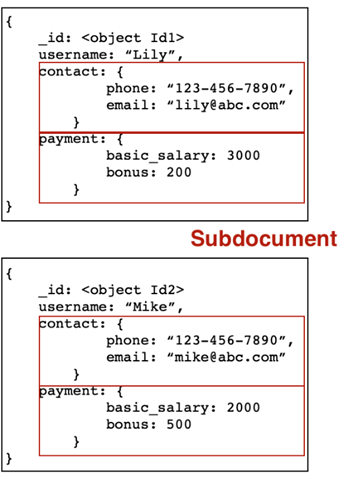
The key components of MongoDB Architecture are shown as following:

1. **Database** – Database is a container for collections similar in RDMS.
2. **Collection** – Collection is a group of documents (like table in SQL), which is stored in database.
3. **Document** – A record in a MongoDB collection is called a document (like row in SQL). The document, in turn, will consist of field name and values.
4. **Field** – Field (like column in SQL) is a name-value pair in a document. A document has zero or more fields.
5. **\_id –** \_id is a required field in every documents. The \_id field is like the document’s primary key which has a unique value. If a new document is created without an \_id field, MongoDB will automatically create the field.

Here is a toy example of the structure of a MongoDB. This is the definition of an employee collection:

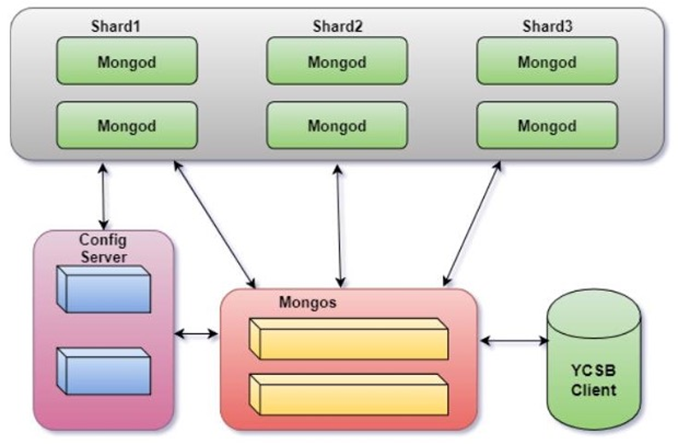


The collection of employee can have multiple documents:



A document can have sub documents, which also has its own implicit \_id in each sub-document.

The general architecture of MongoDB is shown as following:



**Shards**: each shard contains a part of the cluster’s data.

**Config Servers**: contains sharding metadata which including routing information. Config server consists of Replica Set.

**Client**: connect to a mongos, then mongo dispatch client’s request to shard and get the response.

**2.2 Overall strengths and weakness**

There are several typical strengths which differentiates MongoDB from other DBMSs.

**Strength:**

**Easy and intuitive to build and maintain – document-oriented:**

MongoDB is a document-oriented and schema-less database, which means that there are not restrictions in the format and structure of data storage. This a huge benefit for retaining existing data in massive volumes and different structural states. Also, it is easier and faster to create or maintain the database. The document-based database is more intuitive compared to RDBMS which need schema design and normalization.

**Easy and quick to access data – ad-hoc Queries:**

In many cases, we have no idea what queries the end users will run, while designing a database schema. Ad-hoc queries which are opposite to the structured queries, are short-lived commands whose value depend on a variable. For simple and small databases, this advantage is not obvious. However, when it comes to big data project, the ad-hoc query shows its advantage that it is easy and quick to access. Therefore, it can be used in real-time analytics.

**Easy to scale – sharding:**

Sharding is the process of dividing the data into subsets which is called shards. This feature enables horizontal scaling of a database. Each shard can be regarded as a distributed and separate database. Combining the data of the shards and a comprehensive database will provide feasibility for large and growing database.

**Good for time series data – time series data:**

MongoDB provides several features to support time series data. The time series collection in MongoDB is efficient in storage.

Good ecosystem maturity – community support:

MongoDB is open source and have a large and mature community to support developers.

**Weakness:**

**High Memory Usage**:

MongoDB is designed to store key names for every value pair. Because it does not have join function. So, there will be some data redundancy.

**Limited Data Size**:

MongoDB allows a maximum document size of **16MB**, which could be not enough in some situations.

**2.3 Best use cases**

One of the most important use case of Mongo DB is big data. Big data usually refers to the data whose amount is extremely large. Usually, big data are updated frequently. Sometimes these kinds of data are computationally complex to be processed by traditional, hierarchy-based data processing software. MongoDB’s NoSQL, document based, and non-relational structure is perfectly suited for handling big data.

Current there are a lot of big companies using MongoDB, such as Netflix, eBay, LinkedIn and so on. The main reason these big names use MongoDB is because it is a proficient tool for managing large scale of data. It best fit the following scenes:

1. Large scale data
2. Data amount growing rapidly, no downtime while the application is being scaled
3. Large amount and various queries

Specific using scenarios include internet of things, real-time analytics, transaction applications, game industry and so on.

Take China Eastern Airline for a case study. As one of the three major airlines in China. China Eastern Airline has more than a thousand flights and 260,000 passengers every day. In the past, customers ordered a ticket according to an average of 200 database queries. However, nowadays it takes more than 12,000 queries before an order occurs. As the query has increased by a hundredfold, the airline company must support 1.6 billion requests per day.

After evaluation, Eastern Airline chose MongoDB, why?

**2.3.1 From Business Manager**

As a business manager, he/she would evaluate the business need, such as what exact kind of product/service will provide to customers also total cost should be under the company’s budget.

Business manager from Eastern Airlines set the requirements, which provides users with personalized flight search services, supports multi-destination search, search based on budget range, city search, real-time low-price calendar search, etc. At the same time, flexibly combine transfer routes to increase the coverage of OD route.

**2.3.2 From Technical Architect**

After receiving the manager requirement details, technical architect from Eastern Airline set the tech requirements as below:

1. Memory-based query database, reducing disk IO interaction.

2. Complex and changeable data storage structures and types.

3. Cluster architecture to ensure high availability.

4. Complex SQL queries, such as searching for eligible flights based on budget, or searching for eligible flights based on flight time (morning and afternoon).

After evaluating the above requirements, they chose MongoDB.

**2.3.3 From Developer**

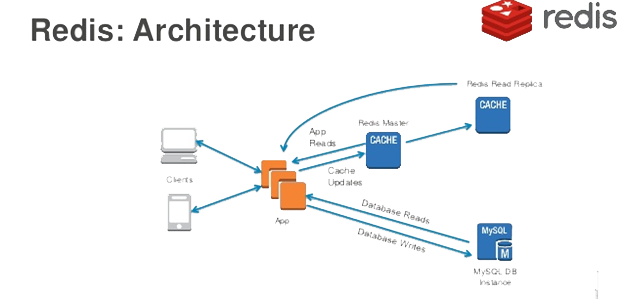
Developer will test the compute process, processing and response time, storage in using MongoDB and finally chose MongoDB.

## 

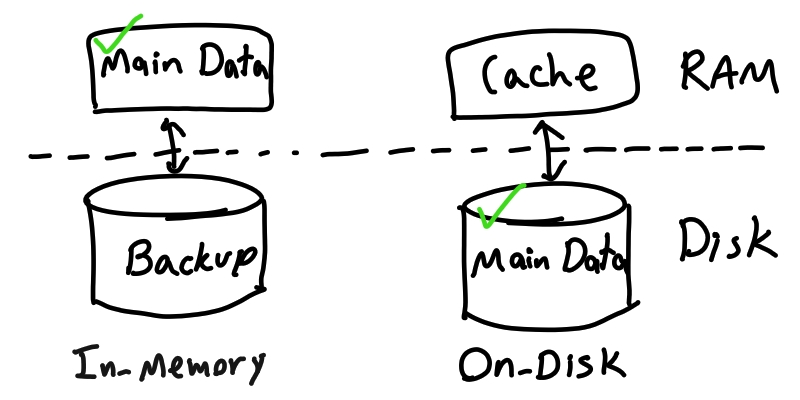
## **Redis**

Redis is an open source NoSQL database, it is very popular, its data structure is in-memory, in the cache, not at storage disks. You can treat it as cache, or message passing agent. Redis support multiple data structures. The applications includes strings, encrypt hash, list, tupil, group, sets, geolocation coordinates, etc. Radius’s in-memory database has faster speed than other NoSQL databases. It requires more RAM but less disks. Redis Cluster allows users to scale up. Because it only allows primary key access, the query function is limited. But the primary key offers much faster speed than MongoDB. If there is complicated query, Redis is not recommended.

### 3.1 Redis Architecture

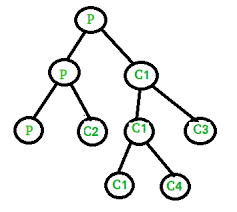
****

Compared to database management systems that use [disk storage](https://en.wikipedia.org/wiki/Disk_storage) systems. In-memory databases are faster than disk-optimized one. disk access is slower than memory access. Accessing data in memory reduces [seek time](https://en.wikipedia.org/wiki/Seek_time) in querying the data, then comes more predictable performance than disk. Redis is short for Remote Dictionary Server, it is an in-memory data structure store, used as a distributed, in-memory key–value database, cache and message broker.



Redis can be considered as a store and a cache at the same time. It was designed so that data is always modified and read from the main computer memory, but also stored on disk in a format that is unsuitable for random data access. The formatted data on disk is only reconstructed into memory once the system restarts. That is why it is referred to as Dictionary server.

Compared to a relational database management system, Redis user commands do not describe a query to be executed by the database engine but rather specific operations that are performed on given abstract data types. Therefore data must be stored in a way which is suitable later for fast retrieval. The retrieval is done without help from the database system in the form of secondary indexes, aggregations or other common features of traditional RDBMS. The Redis implementation makes heavy use of the fork system call, to duplicate the process holding the data, so that the parent process continues to serve clients while the child process creates an in-memory copy of the data on disk.



### 3.2 Supported languages

Many programming languages support the Redis language on the client side.3.3 Data types

Redis maps keys to types of values. An important difference between Redis and other structured storage systems is that Redis supports not only strings, but also abstract data types:

Lists of strings/ Sets of strings (collections of non-repeating unsorted elements)/ Sorted sets of strings (collections of non-repeating elements ordered by a floating-point number called score)/Hash tables where keys and values are strings/HyperLogLogs/ Stream of entries with consumer groups, allows you to store multiple fields and string values with an automatic, time-based sequence at a single key/ Geospatial data /JSON /Graph /Time series data types

### 3.4 Performance

When the durability of data is not needed, the in-memory nature of Redis allows it to perform well compared to database systems that write every change to disk before considering a transaction committed. Redis operates as a single process and is single-threaded or double-threaded when it rewrites the append-only file. Thus, a single Redis instance cannot use parallel execution of tasks such as stored procedures.

Redis typically holds the whole dataset in memory. Dataset can be asynchronously transferred from memory to disk at regular intervals. Alternatively by journaling, a change journal that modified the dataset is added to an append-only file. By default, Redis writes data to a file system every 2 seconds and more, but flexible if needs more configuration. Only a few seconds of data would be lost at the time of system failure.

Master–replica replication mode support many replicate from any server. A replica can become a master to another replica. The copies are single rooted in replication tree system. The publish–subscribe mode allows a client to subscribe to a channel and receive a full feed of messages published to the master anywhere. Replication is good tool for data redundancy.

### 3.5 Use cases

Twitter are using Redis, Amazon Web Services offers ElastiCache for Redis-a managed Redis service, Microsoft also offers Azure Cache for Redis in Azure.

### 3.6 Potential weakness

A potential technical hurdle with in-memory data storage is the volatility of RAM. Specifically in the event of a power loss, intentional or otherwise, data stored in [volatile RAM](https://en.wikipedia.org/wiki/Volatile_memory) is lost.With the introduction of [non-volatile random-access memory](https://en.wikipedia.org/wiki/Non-volatile_random-access_memory) technology, in-memory databases will be able to run at full speed and maintain data in the event of power failure. The re-booting Redis database can recover the data store from its last consistent state.

### 3.7.1 From Business Manager

### 3.7.2 From Technical Architect

### Worth to know — DragonflyDB, a new in-memory database

“Dragonfly is a modern in-memory datastore, fully compatible with Redis API. Dragonfly implements novel algorithms and data structures on top of a multi-threaded, shared-nothing architecture. As a result, **Dragonfly reaches x25 performance** compared to Redis.” - source: Internet

From this news, it seems like DragonflyDB can take advantage of all the cores available on the machine. It shows great multi-threaded throughput. But short mentioning other functions that Redis is good at. We need to wait and research more and find out the new benchmark testing result when they come online

## 

## **Oracle NoSQL**

## **Overview of Oracle NoSQL**

## NoSQL Database represents a continuously evolving enterprise application architecture of the past 20 years, and Oracle NoSQL is a high-demanding solution today among all database products. It builds applications using document, columnar and key-value database models. It is a fully managed database service for developers to focus on application development and not need to deal with the hassle of hardware and software infrastructure. It produces and consumes the data at high volume and velocity and delivers predictable single digits with instantaneous response time to match user experiences. It is 100% compatible with the on-premises Oracle NoSQL Database. The service is built with continuously evolutional data models and offers ACID transactions, which is the acronym for atomicity, Consistency, Isolation, and Durability. Oracle NoSQL offers low pay-per-use pricing for on-demand and provisioned capacity modes, and it scales on-demand based on dynamic workloads.

## The Oracle NoSQL Database is a sharded system that distributes data across many clusters. The number of clusters can increase depending on the size of the business or demand. Within each shard is data replication for high availability. Oracle NoSQL Database failovers the node failure rapidly once found, and optimal load balancing of queries. The language NoSQL Database deploys to build up applications are Python, Node.js, Java, C, C# drives and REST API.

## **Architecture**

## Oracle NoSQL Database reads and writes data by performing network requests through its data store called KVStore. KVStore is comprised of Storage Nodes (SNs), and each of them contains Replication Nodes (RNs). Each storage node is a virtual machine with its local storage. Data is automatically spread across replication codes through the internal KVStore mechanism.

## The following architecture illustration shows the Oracle NoSQL with three main sections and organized from top to bottom. Load balances and web servers, which occupy the top level, manage three application servers. Each of the application servers runs a NoSQL Database application. A Traditional Backend Database is at the next level, connected to one of the app servers. Most of the architecture is in the lower right section, which is the diagram of KVStore. The KVStore includes nine storage nodes and each storage node host one replication code. Nodes will arrange in clustering, and three storage nodes will be responsible for the admin processes. It is recommended to have more than one admin for the KVStore, such as backup. After the network is connected to one of the admin process, administrators and authorized users can invoke utilities and the Administrative Command Line Interface (CLI) to perform set up and other tasks.

## 

*Figure. The Architecuture of Oracle NoSQL Database*

**Strength**

Data Model Flexibility

Oracle NoSQL Database Application does not have a predefined set of data types, making it more competitive than a relational database management system.

Capability to handle the increasing amount of traffic

Oracle NoSQL Database Application can process more queries than other database applications in the same amount of time, even if the amount of traffic is vast, but the response time only increases modestly.

Data-model simplicity

Oracle NoSQL Database is not limited to a table structure; it allows for unstructured data and is simple to query.

**Weakness**

Limited built-in analytical functions

There are fewer analytical functions to choose from in the Oracle NoSQL Database compared to the Oracle database.

Eventual data consistency

It may take time to make the subsequent queries visible after writing or deleting queries.

Data Redundancy

As Oracle NoSQL Database allows unstructured data and no pre-defined data type, and there are no mechanisms to ensure data integrity, users may have redundant data across documents.

**Best Use Case Fit**

1. Online display advertising

The NoSQL Database will be leveraged to record users’ behavioural segments, increasing the probability of clicking on an advertisement. It tends to be high velocity and low latency requests, even if there will be eventual consistency, but it is acceptable in this case. Data will be stored in the cluster and scale with increasing data.

2. Fraud detection online

Oracle NoSQL Database has an instantaneous response time, and the online fraud model needs real-time analysis of large volumes of historical and live data, and the data may be of all types.

## **Business Manager**

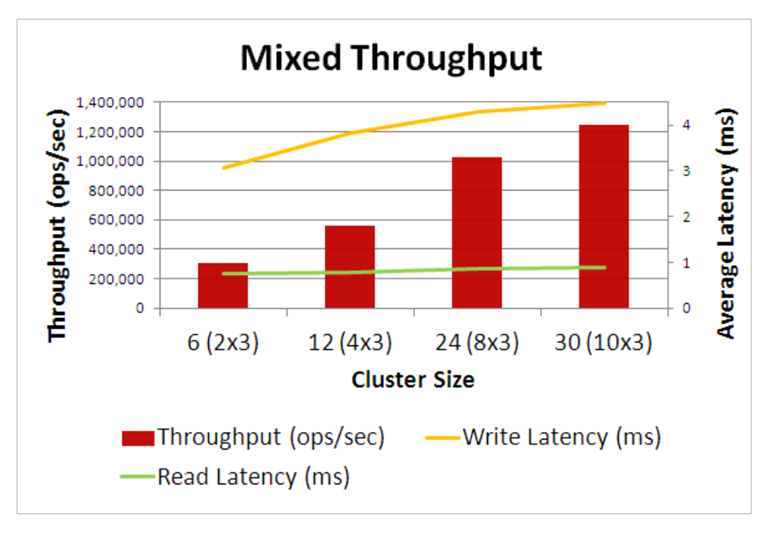
## We have experimented Oracle NoSQL Database configurations to evaluate the performance. Specifically, the system's performance scales with the number of nodes and the upper end of the scale in mixed operations. In the first test, constant data was loaded per storage node to the configuration of varying sizes. The following graph presents the performance of inserting raw data ranging from a single-shard system only with three nodes storing 100 million records to a system with 32 shards on 96 nodes storing 2.1 billion records. The blue line delivers the result of the throughput in operations per second, and the red line demonstrates the response time in milliseconds. It proved that when the system keeps scaling tends to linearly as the database size grows, and the response time only rises at a modest rate.

## 

## Furthermore, given the number of shards and database size growth, the following graph presents the throughput and the response time when the workload with 50% reads and 50% updates. The throughput scales almost linearly while the update and reads latency decrease. It offered the scalability which is sought by today’s demanding applications.

## 

## The second test it aimed at observing the upper end of the database performance scaled as the cluster size grows and with a stated target of 1 million mixed operations per second. The experiment comprised 15 storage nodes, each a 2.9GHz Xeon E5-2690 dual-socket machine with eight cores and 193 GB memory. All 15 nodes are configured into ten shards, with 30 replication nodes in them. 2 billion records were tested. The following graph delivered the result of 1.24 million mixed operations per second and 0.88 milliseconds of average read latency and 0.4 milliseconds of average update latency. Overall, these two tests demonstrated that Oracle NoSQL could meet the peak performance needs of any demanding application even with increasing traffic. It also brings enterprise-quality storage to the NoSQL environment with highly-availability and widely-distribution. Besides the outstanding performance, its feature of fail-over of the nodes has also proven its reliability in that the system can continue to run and keep available after any failure. Moreover, Oracle NoSQL Database will also allow you to integrate with different products in Oracle, such as Cloud Services, and you will be able to retrieve all the information and data anywhere. Therefore, it will enhance work efficiency.



## **Technical Architect**

## The job responsibility of a technical architect is to assist the company in meeting its goal in terms of defining, building, and managing the company’s technology architecture for its information assets. The most important duties include designing, security, backing up/recovery, and monitoring. The Oracle NoSQL database offers the function of automatic encryption of the entire database. Meanwhile, it will also patch and upgrades automatically without any interruption of database operation, it maximizes the security level of the entire database management system; usually, weak security is a weakness of a NoSQL database management system and Oracle NoSQL Database system turns this into an opportunity and lead its database system to be more competitive. Oracle’s always-on encryption function also provides data preservation at rest and in motion. The specific feature of failover over the nodes also ensures the database management system keeps running smoothly and ensures the user experience of Oracle NoSQL experience.

## 

## 

## **Database Developer**

## Data in the real world are anonymized and randomly generated. It is stored in the way of velocity, volume, and variety, which resulted in many large technology companies developing their NoSQL issues as the relational database system could not keep up. Specifically, Google released the Bigtable paper, the foundations for HBase NoSQL data store and GCP’s Cloud Bigtable. Amazon offered Dynamo paper as their alternative solution. Since then, the NoSQL database system distribution has kept developing. Their main priority was to address scalability, which has been an issue for relational database management systems. In RDBMS, the response time is significantly increased once the data volume is raised. Companies must purchase more servers to resolve the issue. However, in Oracle NoSQL, data is stored in the nodes and distributed across the clusters. Companies simply only need to increase the number of nodes to address it. Secondly, data generating different formats is another issue in the traditional RDBMS besides the increasing data volumes. There is no pre-defined data type in Oracle NoSQL which provides developers with more flexibility, and users also have authorization and access to integrated the tools from Oracle and therefore improve the user experience.

## **Conclusion**

NoSQL databases are still evolving and more number of enterprises is switching to move from the traditional relational database technology to non-relational databases. But given their limitations, they will never completely replace the relational databases. The future of NoSQL is in the usage of various database tools in application-oriented way and their broader adoption in specialized projects involving large unstructured distributed data with high requirements on scaling. On the other hand, an adoption of NoSQL data stores will hardly compete with relational databases that represent reliability and matured technology.

NoSQL databases leave a lot work on the application designer. The application design is an important part of the non-relational databases which enable the database designers to provide certain functionalities to the users. Hence a good understanding of the architecture for NoSQL systems is required. The need of the hour is to take advantage of the new trends emerging in the world of databases – the non-relational databases. An effective solution would be to combine the power of different database technologies to meet the requirements and maximize the performance.

Although MongoDB is one of the most popular NoSQL databases, wide column databases like Cassandra may be able to deliver better query performance. When choosing your NoSQL database, you should consider the availability of managed DBaaS services, where you can offload database maintenance and management to the provider. This allows the developer to focus on the application. In this area, HBase is lacking, while MongoDB offers very mature DBaaS offerings, like MongoDB Atlas. HBase is a good solution for write-heavy applications and massive amounts of records.