WHITE PAPER

IMPLEMENTING NOSQL IN ENTERPRISE APPLICATIONS

Abstract

With the advent of web, digital commerce, social computing, etc., there has been a large volume of uncontrolled, unstructured, and information-oriented data in enterprise applications. All enterprise applications do not always require RDBMS to store and retrieve data as the core characteristics of RDBMS do not always support the nature and usage of such data. NoSQL databases make it simple and easy to store and retrieve different types of data in any format.

This white paper examines why NoSQL is a better and future-ready option compared to traditional databases. It discusses some of its key capabilities such as the ability to handle data formats, integrate with legacy systems, horizontal scaling, and more. It also lists out some best practices that can help enterprises develop a business case for migrating to NoSQL databases.

Infosys®
Navigate your next

Introduction

Relational databases have been used by enterprises for decades and have worked well primarily because the focus was on storing and analyzing structured data while keeping storage costs low. The characteristics of data and its applications, however, have changed in recent years.

For decades, most of the data stored in an organizational database was structured. Data needed to be generated and accessed in a controlled manner, and acted as 'records' of business transactions. Such data was always modeled, stored, and accessed using relational database management systems (RDBMS). Such data needed to pass the ACID test of atomicity, consistency, isolation, and durability.

Today, all enterprise applications do not need to meet the stringent requirements of ACID properties, particularly consistency and isolation. Moreover, unstructured data is growing at twice the rate of structured data. Analysts estimate that 80% of enterprise data is unstructured, consisting of images, audio, data feeds, and social media data. Further, enterprises are flooded with big data and want to position services and analytical applications based on big data. Users are looking for up-to-date and actionable knowledge from big data.

Understanding it all requires new approaches that are not possible with relational databases. The inability to analyze all data – structured, semi-structured, or unstructured – can lead to an enterprise missing out on critical information and insights. This can result in the business losing time to market, customers, and revenue.

Handling big data requires advanced technologies for storing data characterized by variety, volume, and velocity. NoSQL databases are fast gaining attention due their specific design and ability to manage big data.

Characteristics of NoSQL

NoSQL databases hold a prominent place in the enterprise data architecture strategy. Given that traditional relational databases cannot deliver the agility, flexibility, and performance that modern enterprise applications need, NoSQL has become critical for building cloud-native applications. Some of the characteristics of NoSQL are:

- **Logical data model** – Logical modelling of data using loosely typed extensible data schema (map, column family, document, graph, etc.) instead of modelling data in tuples
- **Data distribution model** – These databases are designed for horizontal scaling through a data distribution model across multiple nodes. They also abide by the principles of CAP theorem, i.e., Consistency and Availability (as provided in HBase and MongoDB) and Availability and Partitionality (as provided in Cassandra)
- **Data persistence** – Data can be persisted in disk (as in MongoDB and Riak), memory (as in Redis, Couchbase, and Terrastore), or in a combination of disk and memory (as in HBase, Hypertable, and Cassandra)
- **Interfaces** – These databases support various non-SQL interfaces for data access. For example, REST, MapReduce, and language-specific APIs. Further, databases like Couchbase (N1QL) and Cassandra (CQL) provide SQL-like interfaces in their databases

Rationale for Adopting NoSQL Databases

There are many reasons for enterprises to choose NoSQL over traditional SQL databases. As of now, several large enterprises are transforming their environments by running major parts of their business on top of NoSQL databases. The reasons for adopting NoSQL are detailed below:

- **Scale-out strategy and zero downtime** – NoSQL databases uses this strategy to give a clear path for scaling huge amount of traffic. This architecture also provides zero downtime in upgrading a database or changing its structure. It handles large volumes of traffic in efficient way.
- **Ability to handle different structures of data** – NoSQL databases are good choice for storing and modeling structured, semi-structured and unstructured data in one database. They store data in a form that is identical to the objects used in applications.
- **Accelerated development** – As NoSQL databases follows the structure of the data, NoSQL databases are suitable for agile development practices based upon frequent code changes or any other frequently used criteria.
- **Affordable data services** – Big data handling and management is main portion of NoSQL architecture. Data scalability is easily achievable. As NoSQL databases follows a scale-out strategy, and which scale to huge data volumes more cost effective as compared to SQL databases.
- **Support for new application patterns** – NoSQL database are easily adapted to cloud technology. NoSQL databases supports microservices architecture in easy way. NoSQL databases integrate well with real-time streaming technologies. Transactional and analytical capabilities are both supported by a single NoSQL database.
- **Open-source options** – Many enterprise applications lack choice when it comes to open source solutions. Usually, a few general options exist. The availability of skills and local in-country support is another challenge. Enterprises that leverage NoSQL can access end-to-end commercial support such as new features or rolling upgrade through global offices. This reduces the cost of adopting NoSQL technology, and allows prospects to 'try before you buy'. The availability of open source alternatives is also driving healthy competition, nudging commercial NoSQL companies to offer free, rich-feature versions of their software or special start-up licenses to small organizations.

Benefits of NoSQL Databases

NoSQL databases offer several advantages over SQL databases. Some of these benefits are:

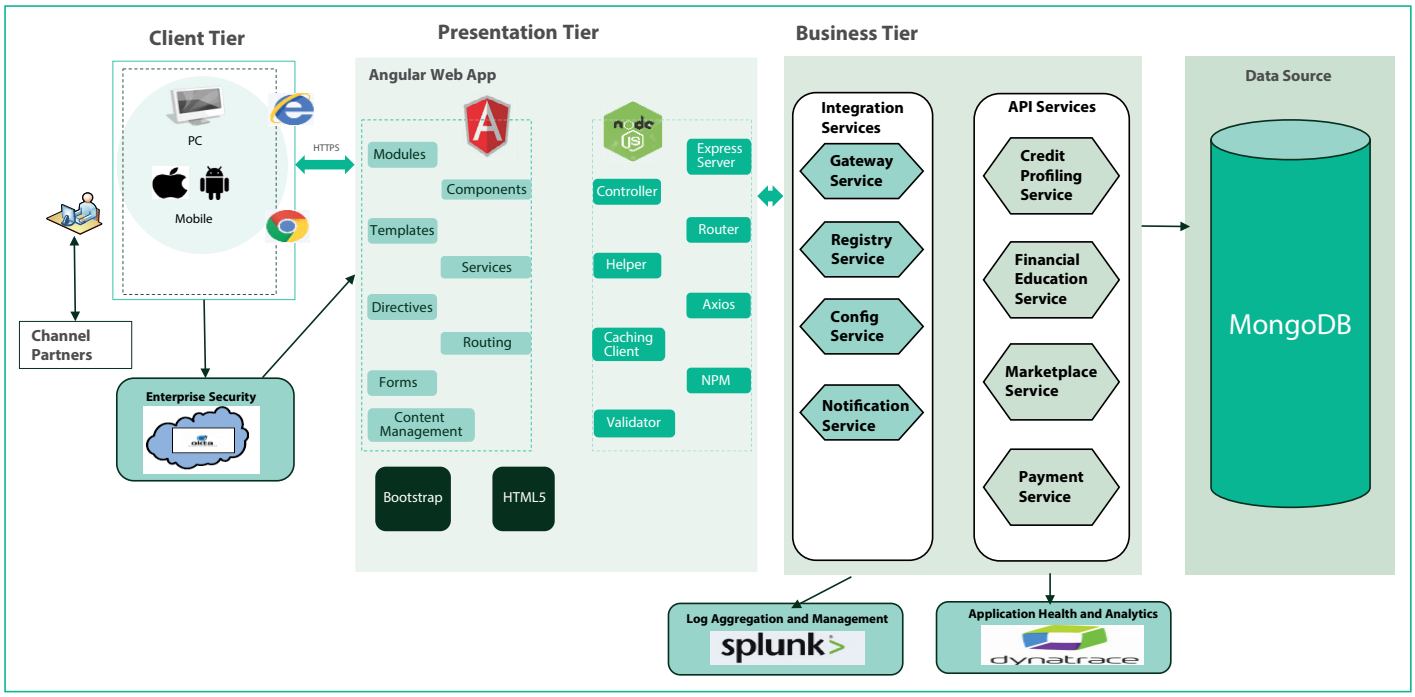
- **Horizontal scaling** – After SQL databases surpass the capacity requirements of the current server, they require vertical scaling which requires migrating to bigger and more expensive server. Conversely, NoSQL databases permit horizontal scale-out where enterprises can add inexpensive servers whenever needed. NoSQL offers auto-scaling and failover capabilities and can be executed in a distributed manner.
- **Faster querying** – As compared to SQL databases, queries execution is much faster for NoSQL databases. SQL databases have normalized data hence queries for one object needs joining data from more than one tables. Due to the size of the table grows, joins become quite expensive. However, SQL databases stores the data in such a way that data is available in optimized form for queries. One principle of NoSQL database is 'Data that is retrieved together should be stored together.' If data is stored in such a way that queries won't require joins then, querying is much quicker.
- **Adjustable data models** – NoSQL databases generally have very adjustable schemas. A flexible schema allows the database to change as the requirement changes. Users can quickly iterate and constantly merge new application features for higher and faster value.
- **Ease of development** – Many NoSQL databases such as document type databases (MongoDB, Couchbase, and DocumentDB) associate their data structures to most used programming languages. This association of data allows users to store data in the same form as they use in their application code.
- **Support for all data forms** – The majority of data in enterprise systems is unstructured. Many NoSQL databases can handle indexing of unstructured text either as a native feature or as an integrated set of services including Solr or Elasticsearch. Entity enrichment services such as SmartLogic, OpenCalais, NetOwl, and TEMIS Luxid combine extracted information with other data, thereby providing rich information and enhancing analytics effectiveness and use.
- **Agility without legacy** – NoSQL databases are fairly new and, hence, have either no or minimal legacy code. There is no need to provide support for old hardware platforms or to keep updating obscure functionalities.
- **Reduced need for Extract, Transform, Load (ETL)** – NoSQL databases support storing data as-is. Key value storage stores simple data structures whereas document NoSQL databases can handle a range of flat or nested structures.
- **Support for multiple data structures** – Some applications need simple object storage whereas others require highly complex and interrelated structure storage. NoSQL databases support a range of data structures. Related information values can be grouped in column families within Google Bigtable clones. Many data structures like strings, maps, lists, simple binary values can be handled at much higher speed in key value stores. Highly complex parent-child hierarchical structures can be managed within document databases. A web of interrelated information can be described flexibly and related in triple and graph stores.
- **Transactions and consistency in NoSQL** – Some of the NoSQL databases allow users to specify the level of consistency for some of the operations while some of the NoSQL databases even fully support ACID transactions.

  But some of the NoSQL databases compromise consistency in terms of availability, scalability, and partition tolerance. However, for some of the document store NoSQL databases or key-value NoSQL databases, transaction consistency is not much needed as most operations are atomic.
- **Data access** – NoSQL offers easy-to-use interfaces for storing and delivering queried data. APIs allow low-level data manipulation.

Case Study

A financial services company faced certain challenges with its relational database management system. They evaluated migrating to NoSQL and embarked on a comparison of the two database models as described below:

- **Cost savings** – The existing database was costlier, licensed, and used larger servers and storage systems. The proposed NoSQL database would use less expensive servers to manage rising data and transactions, in addition to being open source.
- **Big data management:** The RDBMS lacked proper provisions to manage big data flowing in real-time across the company's systems. This was a key challenge for process models where applications need constant, faster, and high-volume data feeds. NoSQL showed the way to manage big data by storing larger volumes with more efficiency.
- **Scalability** – Scalability in RDBMS involves hardware enhancements, which was proving costly. Relational databases are centralized and follow a 'share-everything' approach. NoSQL databases are distributed in nature and follow scale-out technology. Scalability is assured with node-based cluster architecture that can manage load on the fly, a key requirement for big data applications.
- **Relational data model fitment** – The relational data model does not integrate with every domain. Using a relational database to traverse the graph can become extremely challenging due to the number of 'table joins'. On the other hand, NoSQL databases have a flexible data model that supports changes to the database as per the requirement.
- **Performance hit** – For relational databases, performance is impacted due to joins, ACID transactions, and strict consistency constraints (particularly in distributed environments). Queries in NoSQL databases are very fast than in SQL databases. Data storage is done in NoSQL databases by keeping optimization of queries in mind.



With NoSQL emerging clearly as a better option, this financial client chose to migrate to MongoDB. This enterprise application uses Angular and NodeJS for front-end UI components and microservices architecture to segregate the business functionality in different scalable microservices. Following its implementation, the NoSQL solution has accelerated response time and enabled dynamic scalability.

MongoDB is a NoSQL database program that leverages JSON format to store data. Its scale-out architecture helps distribute workloads in smaller nodes, allowing enterprises to better handle spikes in traffic as the business grows. It also supports database transactions that allow multiple changes to a database to be grouped together and either made or rejected in a batch. This transaction feature of MongoDB is particularly useful for financial organizations.

For this particular client, the application required nested structures with variations in data, making MongoDB the best-suited database. It supports a large number of transactions and promotes out-of-the-box scalability.

Implementing MongoDB has helped the financial services organization achieve:

- Reduced latency
- Rich and faster querying
- Flexible indexing
- Aggregations in database
- Known data relationships
- Ability to work with any subset of data
- In-memory caching of results

Best Practices

Some best practices that can be followed for NoSQL databases are:

- ***Use key-value stores*** for the persistent sharing of data by multiple processes or microservices in an application
- ***Leverage multiple deployment options*** to host NoSQL databases on Amazon EC2. Options include 'single region and multiple availability zones' or 'multi-region deployment'. With the help of AWS Regions and Availability Zones, Amazon EC2 offers multiple deployment options that provide highly available workloads
- ***Duplicating data attributes*** across different tables or collections can be very useful, in some cases, to avoid complex queries
- ***Encrypt sensitive database fields*** as well as any information that is subject to compliance regulations. In case of incidents, NoSQL database must have access to the back-up and disaster recovery
- ***Handle transactions and rollback at the application level.*** Most NoSQL databases provide some foundational features for user conveniences while others support features such as conditional writes and atomic updates to implement transactions efficiently
- ***Organize data as documents*** with trees to mirror how people think about information, i.e., hierarchically and in parts and sub-parts around a single topic. With documents, relationships are inherent and document models are inherently 'de-normalized'

Conclusion

Fortune 500 companies like Facebook, LinkedIn, and Twitter are adopting NoSQL to serve millions of users across the world. NoSQL databases process unstructured data easily and dynamically using a flexible and cloud-friendly approach. These databases leverage a scale-out model to add new servers to the cluster. Optimized specifically for applications that require large data volume, low latency, and flexible data models, NoSQL databases are now increasingly being used for big data analysis. They also offer real-time decision-making on large datasets.

NoSQL databases provide many benefits like consistency, availability, and partition tolerance. They promise a high level of scalability, are easy and cost-effective to implement, and provide faster access to data. With the real-time analytics, NoSQL also supports fraud prevention and financial crime monitoring. Shifting to NoSQL databases can help enterprises accelerate innovation and meet customer demands and expectations.

About the Author

Mahesh More
Senior Technology Architect, FSSTAR, Infosys

Mahesh is a Senior Technology Architect with Infosys, with more than 17 years of experience in Java\J2EE. He is involved in developing new digital architecture software solutions, and advises clients on the technical and business value to be achieved through technology innovation. Mahesh is responsible for reviewing, analyzing, and evaluating market requirements, business requirements, and project briefs in order to design the most appropriate end-to-end technology solutions.

References

1. https://aws.amazon.com/nosql/
2. https://www.infoworld.com/article/3240644/what-is-nosql-databases-for-a-cloud-scale-future.html
3. https://www.ibm.com/cloud/learn/nosql-databases
4. https://www.geeksforgeeks.org/introduction-to-nosql/

For more information, contact askus@infosys.com

Infosys®
Navigate your next