|  |
| --- |
| Databases 4 |
| MongoDB Report |
| The Implications of selecting a NoSQL rather than a relational database |

|  |
| --- |
| Alicia Smith | A00267161  11-22-2021 |

# Background

The relational model refers to a data model which represents data in the form of tables; each record in the table contains a primary key and the attributes of each record are represented in the columns of the table - this format simplifies establishing relationships between data points. Relational models have many key benefits, primarily the implementation of normalization and schemas; the normalization of relational databases mean that data is only stored once in one location, this is crucial in the reduction of data redundancy the prevention of data anomalies. The use of schemas in relational models can be restrictive in the management of Big Data such as internet usage data, however schemas have an important role in a variety of use cases, the idea being that users can specify exactly what information each record within a table and each table within a database will contain, therefore maintaining a strict structure and avoiding inconsistencies. Relational databases are particularly useful for use cases such as stock management, customer account information, and student academic statistics.

NoSQL started it’s popularization in 2009 in response to the need for processing large volumes of unstructured web data, or Big Data. The NoSQL model is based on a distributed database system, as it is horizontally scalable in nature, it was quickly adopted by large multinational technology conglomerates – such as Facebook and Google - as a quick, low-cost way to process huge amounts of unstructured data. On a smaller scale, NoSQL databases have grown in popularity for smaller use cases as users have the ability to query data without having to learn SQL, making it a good choice for users and businesses who may not have any prior experience with DBMS.

Despite the obvious benefits of NoSQL models and the shortcomings of relational models, the latter remains dominant across many industries; this can be reduced to the fact that the relational data model suits such a wide range of application use cases. Another reason for NoSQL not yet dominating the market is that NoSQL-friendly analytics tools are still in their infancy and have not yet reached the point of being user-friendly and are therefore not accessible to those without an understanding of data science.

# Comparing Relational and NoSQL Models

There are many fundamental differences between the relational model and NoSQL, the most explicit difference between the two is structure; SQL databases are table-based and designed with relationships between entities in mind, in contrast, NoSQL are not based on relationships and can have varying structures. MongoDB, one of the most popular NoSQL database models, makes use of a flexible schema and data stored within it can easily be restructured; MongoDB stores data as documents in the JSON format and also supports key-value pair concepts.

Scalability is another definitive contrast between the two mentioned database models, Relational SQL models are known to be vertically scalable, meaning that all data is stored on a single server and require more physical and virtual resources to adequately host larger databases. Conversely, NoSQL models are horizontally scalable - data can be partitioned and distributed across multiple servers, this essentially makes NoSQL databases relatively boundless. There are obvious benefits to the NoSQL approach on a corporate level, any truly large-scale operations will quickly become very expensive in terms of hardware in the relational approach, whereas overall system cost is comparatively low when it comes to NoSQL.

# Comparing NoSQL Models

The term “NoSQL” means “non-SQL” or “not only SQL” and is used as umbrella term for any database model which is not based on the relational model, of where there are many; some of the most popular NoSQL DBSMs are: Redis, Voldemort, Cassandra, HBase, MongoDB, Neo4j, and Firebase. While each of these falls into the catch-all of not utilising the relational model, there are several operational models associated with NoSQL. Redis and Voldemort are key-value stores, Cassandra and HBase are columnar databases, MongoDB and Firebase are document-oriented stores, and Neo4j is a graph database model. Although this is not a comprehensive list of NoSQL DBSMs or operational data models, it illustrates the variance amongst non-relational database solutions.

Despite the explicit differences between the models listed, the majority of NoSQL databases share some key characteristics, each of these models strive for more flexibility than the rigid schema defined by relational databases, therefore making all NoSQL models suited to storing unstructured and semi-structured data.

Key-value databases are efficient and highly scalable, they work by storing data as associative arrays that contain key-value pairs in which the key acts as a unique identifier to retrieve the associated data. This data can be simple or complex in nature, however key-value databases are typically used for caching and session management. Comparably, columnar databases are also designed to maximise scalability. Column-orientated databases store data of the same type in columns, similar to the format of the relational models, however these columns are not grouped together in a table but instead stored in separate files.

Document-orientated databases are a type of key-value store, each document stored has a unique key and the document itself is the associated value. This model has grown in popularity in recent years due to their flexible schema, high scalability, and sharding capability. MongoDB is the leading distributed document store globally and is particularly equipped to manage large amounts of unstructured/semi-structured, unrelated, and complex data. Both Firebase and MongoDB are post-relational data models that prioritise speed and flexibility, Firebase is a more apt choice for smaller applications as it can not handle complex querying and has limited sorting functionality, making it more difficult to scale than some of the more popular NoSQL models.

# Conclusion

It is undeniable that both relational models and NoSQL models each have their many strengths and weaknesses and are primarily suited to different use cases, however there is a growing need for multi-model databases and while not all models support this capability, some large companies are beginning to implement both relational and non-relational models to suit their needs. Instagram is an example of a large tech-based company which employs both SQL and NoSQL solutions to it’s backend development. This reinforces the notion that choice of DBMS model can not be concluded on the differences or similarities of the model but only by the project requirements. Despite the rapid growth od NoSQL data models in BigData and other fields, it will never supersede traditional relational SQL databases.