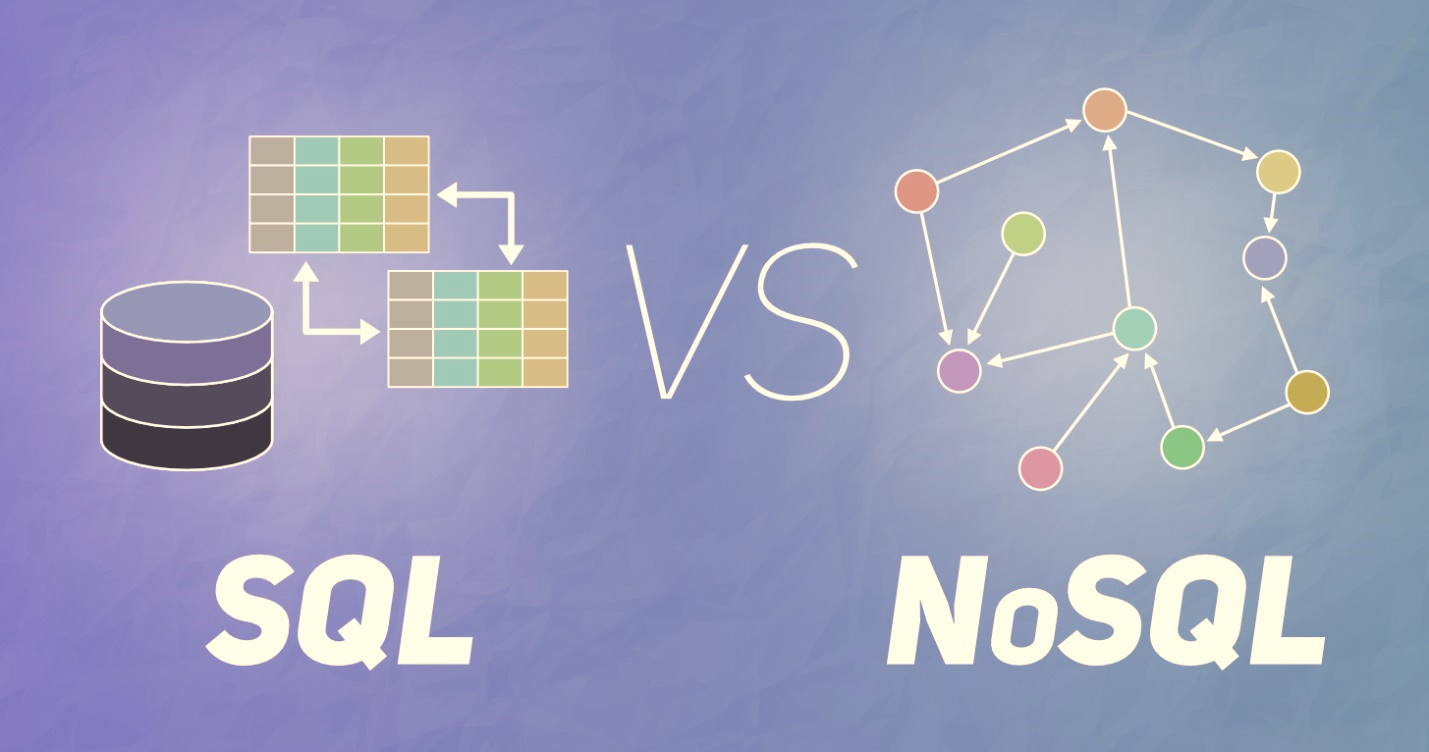
**Final Paper: White Paper Analysis**

**IS 680 Database Management Systems**

**RDBMS Oracle SQL and NoSQL MongoDB**



Alex George

Professor Li

May 13, 2020

**Table of Contents**

Introduction 3

Data Modeling 3

Consistency Model 4

Query Capability 5

User Interfaces 6

Performance 6

Cost 7

Conclusion: My Experience 7

Reference Page 9

**Introduction**

The advent of big data has pushed the importance of how data is best stored and manipulated. Companies are adopting newer database applications that offer increased performance and scalability to provide their customers improved user experiences. Choosing the right database application that meets a company’s requirements is a daunting task. Today’s market provides a conglomerate of database applications. Over the past decade, developers have chosen to cast aside Structure Query Language (“SQL”) as a relic that cannot scale with these growing data volumes. Not Only SQL (“NoSQL”) applications (MapReduce, Cassandra, MongoDB, and more), have become the leading choice for database developers. As non-relational systems such as MongoDB thrive, relational database management systems like Oracle SQL are experiencing a resurgence. In this paper, I will discuss the advantages and disadvantages of MongoDB and Oracle SQL.

SQL is the primary language for accessing and managing a relational database. SQL statements manipulate and query structured data from data organized in rigid table formats. Structured data refers to data that has a logical data model defined by a schema.

NoSQL emphasizes that it may support SQL-like functionalities. Non-relational databases, like MongoDB, store and fetch information from unstructured data. Unstructured data is schema-less meaning the data is organized without an intended structure.

**Data Model**

In a Relational database management system (“RDBMS”), every row in the table represents a collection of related data values. These rows in the table denote a real-world relationship or a logical connection between different tables. These relationships are either a one-to-one, one-to-many, or a many-to-many between tables. Each row in a table has its own unique primary key. The primary key’s main purpose is to identify the relationships between tables. SQL supports entity integrity and referential integrity both of which are fundamental principles of relational databases. These principles actively seek to ensure the quality of data in the database. Relational databases have a fixed structure with pre-defined schema meaning the data types, char, varchar2, number, date, hold constraints that the system must obligate, or a validation error will occur.

|  |  |
| --- | --- |
| Oracle SQL | MongoDB NoSQL |
| Table | Collection |
| Row | Document |
| Column | Field |
| Join | $lookup |

*Data Model Comparison Table*

One of the fundamental changes between SQL and MongoDB is the data model. Relational structures insert data into rigid tubular columns and rows. NoSQL’s denormalized data models allow developers to retrieve and manipulate related data within a single collection. NoSQL databases have dynamic schema meaning the data types int, double, string, array, object, and date, hold no validation constraints. Moreover, documents in each collection may have a different set of fields. The data resides in a collection of key-value pairs, such that each pair is unique and not repeated. \_id serves as the primary keyand is automatically in the collection. However, developers can choose to create their own \_id value. One advantage of having a flexible schema is the ability to embed data into a single collection that would otherwise be multiple parent-child table relationships in a relational system.

Furthermore, unlike Oracle SQL, MongoDB stores JSON documents in a binary representation named BSON. JSON stands for JavaScript Object Notation and developers store JSON objects within MongoDB. Developers access the data through the Mongo Shell or graphic user interface like MongoDB Atlas. MongoDB allows developers to write JSON Schema documents that can determine data type constraints like how RDBMS creates their tables with a pre-defined schema. SQL comes with DML or data manipulation language which is responsible for modifying the data by inserts, updates, deletes, and queries. To further illustrate the differences between MongoDB and Oracle SQL, consider how each manipulates the data:

*Create table Student:*

SQL:

create table student (

studentNumber char(3),

studentName varchar2(10),

studentMajor varchar2(10), constraint fk\_stu\_major references majors(major),

student gpa number(3, 2),

primary key (customerNumber));

NoSQL:

db.createCollection("student");

*Insert data into table Student:*

SQL:

insert into students values ('101','Alex',3,'IS',3.5);

NoSQL:

db.students.insertOne({name: "Alex", major: "IS", gpa:3.5})

*Find students where major = IS and name = Alex:*

SQL:

Select studentNumber

From student

Where Major =’IS’ and studentName = 'Alex';

NoSQL:

db.students.find({major: "IS", name: 'Alex'});

**Consistency Model**

RDBMS uses the acronym ACID to describe the fundamental set of properties that guarantee database transactions are reliable. ACID ensures data remains consistent and accurate even in the event of a failure. ACID stands for Atomicity, Consistency, Isolation, Durability and defined as follows.

Atomic: Everything in a transaction succeeds or the entire transaction is rolled back.

Consistent: A transaction cannot leave the database in an inconsistent state.

Isolated: Transactions cannot interfere with each other.

Durable: Completed transactions persist, even when servers restart etc.

Subsequently, NoSQL is not ACID compliant and relies on a different set of base properties known as the CAP Theorem. The theorem states that a database system can achieve at most two of the three:

Consistency: Every read receives the most recent write or an error

Availability: Every request receives a response

Partition tolerance: The system continues to operate despite any communications break between nodes in the system.

Non-relational systems fail to provide consistency and high availability at once. However, MongoDB 4.0 adds support for multi-document ACID transactions, so it is more user-friendly to SQL developers. The consistency model used by MongoDB, like most non-relation systems, is BASE. BASE is an acronym to describe eventually consistent for non-relational database systems and has three states:

Basically Available: always available for requests.

Soft state: can deal with stale data.

Eventual consistency: eventually it will become consistent if no new changes.

Base properties are less rigid and weaker in consistency than ACID transactions.

**Query Capability**

Queries are nothing more than questions for the database to return a response. RDBMS relies on optimizing storage efficiency since storage was expensive in the 1960s. However, storage has incrementally become more affordable and more efficient over time. Relational models can support the use of complex query capabilities. SQL provides significant value using more complex data access patterns and pushing computational power to the data. As the complexity of queries and entities increases, databases that are ACID compliant become a better option. However, relational models cannot manipulate massive data sets since they must repeatedly check and validate with ACID rules.

Many of the NoSQL systems achieve their primary goal. MongoDB provides a key-value store with a single interface scaled out over multiple machines with high performance and availability built-in. High availability is requirement for companies in certain industries. Companies require high availability must compromise and, filter multiple data properties, or the result will be cumbersome. If query patterns are simple retrieval scenarios, key-value databases are a good choice.

**User Interface**

MongoDB Atlas public cloud web service is a DBaaS or database as a service where administrators can create, deploy, manage, and scale MongoDB databases on Amazon Web Services, Microsoft Azure or Google Cloud Platform. Atlas is user-friendly for developers with adequate experience. Atlas includes a rich collection of data available to developers. Newer developers can utilize Atlas’ graphic user interface to interact with the data. The simple GUI enables administrators to select a cloud provider region and add additional storage whenever needed. Furthermore, Atlas has built-in features that provide industry level security protection, fully managed backups, and automated software patching. Atlas allows developers to use MongoDB Charts to quickly and effectively show visualizations without the need for an additional software provider. Subsequently, if developers are using existing business tools like Tableau there is MongoDB Connector for Business Intelligence. Outside of MongoDB Atlas, developers primarily use the Mongo Shell to perform operations and interact with the data. The shell is an interactive JavaScript interface to MongoDB where developers can paste JSON documents in the shell.

SQL user interfaces is similar to the interface used by MongoDB. SQL or SQL Plus is primarily a command line application where developers interact with the oracle database through a SQL plus command terminal. The SQL terminal enables developers to execute procedures, preform update operations, and query existing data. It is common for developers to write scripts in a text editor and paste those results in the terminal. The SQL language is exceptionally simple to use and understand due to the straightforward terminology such as Select, From, Where, Group by, etc. Additionally, Oracle SQL has one of the largest database community followings such that newer non-relational database applications either create SQL-like query languages or the more popular approach, integrate SQL into the newer application. Oracle SQL comes with a free graphical user interface named SQL Developer. With the GUI developers can browse, create, edit, and delete database objects; run SQL statements and scripts; and view and create reports. Oracle’s graphic user interface gives database developers a convenient way to perform basic tasks.

**Performance**

Scalability and performance go hand in hand, with organizations needing to maintain the access of a vast number of users and data volumes. Scalability references the system’s ability to handle an increase in data growth. While performance indicates the largest possible workload processed while optimizing system resources.

The increasing need for scalability in database designs reveals challenges for RDBMS vendors. Oracle SQL databases are vertically scalable in design. Vertical scaling focuses on increasing the power and capacity of a single server such as using a more powerful CPU or adding more RAM.

MongoDB databases take advantage of horizontal scalability to handle an increase in traffic. In simple terms, horizontal scaling is when data distributes over multiple machines or shards. MongoDB uses sharding to support activities with large data sets. Each machine handles a subset of the work and increases the availability and storage capacity. The sharded cluster can continue to perform read and write operations even when they are not available making a high amount of data availability. Websites that handle high-volume traffic cannot compromise performance and availability due to the lasting effects of an outage. Imagine if Amazon went down for five minutes there would be long-lasting financial and brand image consequences.

**Cost**

A relational database is bound to me or costly than a non-relational system due to the large overhead cost for installing, storing, maintaining, and managing the data in-house. These organizational expenses attribute to paying software licenses, hiring a developer team, and building the data center hardware which includes servers, routers, cables, real estate space, a constant power supply, and an abundance of cold air. To maintain the on-site database administrators must consider the cost of redundant storage, cost of replacement servers, and the cost of maintaining on-going hardware and software. The upfront hardware equipment for storing and accessing the data is not the only expense organizations need to think about. There will be continuing costs of maintenance and support. As well as, the recurring cost of specialized labor, like DBAs and System Analysts. The upfront and continuous maintenance costs is an overwhelming reason why companies choose to look for database alternatives.

A NoSQL database does not suffer the same upfront costs as a traditional system. With MongoDB, users can mitigate the need to employ a developer team, host physical hardware, and fulfill system backups required with RDBMS. NoSQL databases that use sharding significantly reduce database expenses. Sharding is cost-effective since several low-cost machines are often cheaper than buying a smaller number of machines with significantly larger specifications. MongoDB offers a free tier for a single-region cluster limited to the minimal specs of 512 MB of storage. MongoDB’s paid tier is flexible by charging on the size of the cluster. Cluster sizes range from 512 MB of storage with shared RAM to 488 GB storage and 488 GB of RAM. These prices range from less than ten cents per hour to over fifty dollars per hour. MongoDB offers additional paid services alongside the initial cluster including managing backup services, having additional shared clusters, and having faster restores of datasets. Cloud-hosted applications, like MongoDB, are more cost-effective since there is no need for on-site hardware and no need for specialized personnel to manage it.

**Conclusion: My Experience**

The choice between Oracle SQL and MongoDB NoSQL comes down to what database design is most beneficial. I believe both systems are equally useful, and neither dominates the other in capabilities. Oracle SQL was simpler to implement and script because I have two years’ experience with SQL and PL/SQL. I also found MongoDB easy to use with the user-friendly GUI. Since I have more experience with SQL, scripting with JSON was more difficult. However, designing and manipulating the data was a breeze in MongoDB. In my opinion, the system that best fits the organization’s application needs is the system that I would recommend.

SQL’s relational database model is simpler than to MongoDB. The database is easy to design, create, implement, and query because of the inherent data independence and structural dependency rules. SQL’s ACID compliance is powerful enough to support strong consistency. It has a robust query language making it easy to develop elaborate queries. NoSQL databases are feasible when administrators have reached or close to reaching the limitations of an SQL database.

MongoDB might be the better choice for organizations that rely on consumer engagement where latency has a direct impact on the business. MongoDB’s architecture has high performance, high scalability, high accessibility, and is suitable to handle large amounts of unstructured data. MongoDB uses sharding to scale traffic amongst many machines by synchronizing data across servers, racks, and data centers. The schema structure is flexible allowing developers to use any data type without schema validation enforcement. Lastly, having a database as a service significantly reduces operational expenses for most companies by not needing an on-site data center or the specialized personnel needed to maintain it.

Reference Page

An Illustrated Proof of the CAP Theorem. (n.d.). Retrieved April 27, 2020, from  [https://mwhittaker.github.io/blog/an\_illustrated\_proof\_of\_the\_cap\_theorem/](%09https://mwhittaker.github.io/blog/an_illustrated_proof_of_the_cap_theorem/)

Data Modeling Introduction. (n.d.). Retrieved April 27, 2020, from  [https://docs.mongodb.com/manual/core/data-modeling-introduction/](%09https://docs.mongodb.com/manual/core/data-modeling-introduction/)

Kulkarni, A. (2019, July 10). Why SQL is beating NoSQL, and what this means for the future of data. Retrieved from

<https://blog.timescale.com/blog/why-sql-beating-nosql-what-this-means-for-future-of-data-time-series-database-348b777b847a/>

MongoDB Atlas Pricing on AWS, Azure & GCP. (n.d.). Retrieved April 27, 2020, from  [https://www.mongodb.com/cloud/atlas/pricing](%09https://www.mongodb.com/cloud/atlas/pricing)

Saravanan, N. (2020, April 8). Database Scaling: Horizontal and Vertical Scaling. Retrieved April 27, 2020, from

<https://hackernoon.com/database-scaling-horizontal-and-vertical-scaling-85edd2fd9944>

Sasaki, B. M. (2018, November 14). Graph Databases for Beginners: ACID vs. BASE Explained. Retrieved April 27, 2020, from

<https://neo4j.com/blog/acid-vs-base-consistency-models-explained/>

Sharding. (n.d.). Retrieved April 27, 2020, from

<https://docs.mongodb.com/manual/sharding/>