Mongo DB



Paper presentation for

Database Management System (CS 231)

April 28, 2016

**Acknowledgment**

Working on this project “Mongo DB” was a source of immense knowledge to us. We would like to express our gratitude to our instructor Mr. Amit Kumar for his guidance and valuable support throughout the course of this project work. We acknowledge with deep sense of gratitude, the encouragement and inspiration received from our faculty members and colleagues. We would also like to thank our parents for their love and support.

Thank you

Attili Uday U101114FCS053

Dandu Geet Kamal Tej U101114FCS060

Kandula Mohan Sai U101114FCS076

Loga Prakash U101114FCS087

M Andrews U101114FBT170

**Introduction**

MongoDB comes from the word *humongous*. The leaf in the logo of MongoDB has a meaning to it, the founders of MongoDB believe that coding should be natural, and so should using a database. They want the experience of using MongoDB to be simple and natural. Thus the leaf.

There are three types of databases

1. Relational Databases,
2. Online Analytical processing,
3. NoSQL database.

Lot of start-up’s adopt these NoSQL database because it gives them the flexibility in the future to change those things. MongoDB is scalable, open source, high performance and document oriented database. Document oriented database is a class that falls under a broader category called NoSQL database.

It uses dynamic schemas, meaning that you can create records without first defining the structure, such as fields or types of their values. Later it can be changed by simply adding new fields. This model gives us the ability to represent hierarchical relationships, to store arrays, and other complex structures easily. It stores data in JSON like documents that vary in structure. Concepts in MySQL have similar analogy in MongoDB.

|  |  |
| --- | --- |
| **MySQL** | **MongoDB** |
| Table | Collection |
| Row | Document |
| Column | Field |
| Joins | Linking’s |

No tables, no rows, just collection and documentation.

The concept of MongoDB is that all the data is natively joined to where joins are necessary, everything is just one pack of information. It becomes quite handy we can grab one piece of information and automatically have the other, we don’t necessarily have to go off and find other query or another collection to find something else that is related. We filter what we don’t want and reduce the traffic we get from the MongoDB connection. We can only get a subset of data what the document has.

Difference between the queries of MySQL and MongoDB.



MongoDB isn’t actually storing information in JSON it’s supporting something called BSON (Binary Structured Object Notation) which is helping MongoDB internally, which helps in the query performances. Like MySQL, MongoDB has rich set of features, with MongoDB you can also make use of these features across more diverse data types than with a relational database, and at scale.

|  |  |  |
| --- | --- | --- |
|  | MySQL | MongoDB |
| Rich Data Model | No | Yes |
| Dynamic Schema | No | Yes |
| Easy programming | No | Yes |
| Data locality | No | Yes |
| Complex transactions | Yes | No |
| Auto-shading | No | Yes |

**Discussions**

Recently information of over 93 million voters in Mexico has been leaked online. **No password** error in Amazon database AWS which hosted MongoDB database configured incorrectly. No database model has its’ own intelligence to provide security if it’s not provided. It’s not actually the problem with MongoDB, but MongoDB should use search engine transparent database model like oracle.

**Challenges with MongoDB**

MongoDB has a lot of great features like Document oriented storage, Index support, Replication and availability, Auto-shading etc. but it also got some limitations and challenges to it. They are

1. Global write lock problem
2. Low Auto-Sharding reliability
3. Schema-less issues

**1. Global write lock problem**

Global lock is a feature in MongoDB which means that the entire server is locked when a write operation is performed. This is the most oft-cited problem with MongoDB.

**2. Low Auto-Sharding reliability**

Process of sharing data records across multiple machines in order to meet the demands of data growth is approach of MongoDB. As the size of data increases, a single machine may be not be sufficient to store the data nor provide acceptable read and write throughput.

But, MongoDB has an issue with this Auto-Sharding. It cannot keep track of the number of documents present in a collection after Sharding. Uneven shard load distribution is another issue with this Sharding in NoSQL databases.

**3. Schema-less issues**

Schema-less has a lot of benefits it has the ability to change the data rapidly and update to the required new product features. Schema-less also got some cons to it, it has poor integrity and ambiguity.

{

“id1” : ObjectId(“2s1d515sgs5341fg”),

“title” : “DBMS Project”,

“content” : “MongoDB project report”,

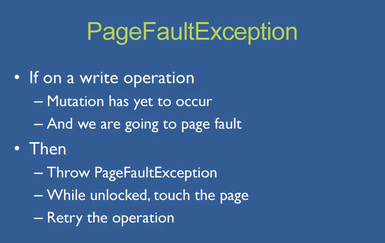
“date” : “28 April 2016”

}

Schema-less is not actually schema less. Every document should contain the schema, which means that the schema has to be repeated.

**Solutions**

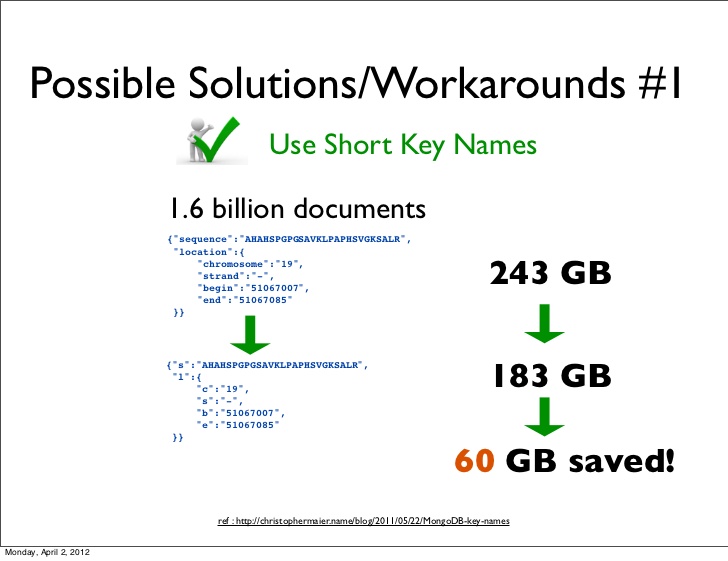
1. Global lock issue is solved in the newer versions of MongoDB. Global lock is eliminated by true database level locking. It requires architectural changes to existing databases. If you are using one large collection it makes only a little difference because you are still using a single database. One collection per Database can be used to reduce lock ratio.



MongoDB v2.2 has rectified the global lock issue with page fault exception technique. It will detect the page fault and touch the page before the mutation has occurred during the write.

2. Auto-Sharding causes the discussed problems, they can be solved in few ways. 10gen the company which owns MongoDB has come up with few solutions that already solved this issue. We should use a system that has automated elasticity and failure handling. Estimating the system requirements and pre planning the required capacity also can help. Have only read option where a system is accessible for reads if it can’t handle writes while repairs are going on.

3. Schema-less issues in MongoDB use short key names which reduces the storage.



MongoDB can use an option to store data compressed. Oracle handles this as transparent to database server at the block engine level. They compress the blocks rather than at a record level.

**Conclusion**

Organisations are adopting MongoDB because it enables them build their applications faster. Most modern applications requires a flexible, scalable systems, applications that require complex, multi row transactions would prefer MongoDB. MongoDB and MySQL both are used in some cases, many e-commerce applications use a combination of both MongoDB and MySQL. MongoDB removes the object-relational mapping that translates objects in code to relational tables. Schema design in MongoDB is more of an art. The two largest benefits to embedding subdocuments are data locality within a document and the ability of MongoDB to make atomic updates to a document. We’ll talk about the pros and cons of MongoDB for final conclusion.

**Pros**

MongoDB can be used if the application requires the following features, Flexible data model, Elastic scalability and High performance.

**Flexible Data Model.** MongoDB databases easily store and combine both structured and unstructured data. Data can also be dynamically updated to evolve with new requirements and without any downtime to the application.

**Elastic Scalability**. Low cost, commodity hardware, allowing for almost unlimited growth.

**High Performance.** NoSQL databases are built for great performance, measured in terms of both throughput and latency.

**Cons**

**No joins:** MongoDB doesn’t have chance for joins like in RDBMS (Relational Database Management Systems). This can be done only by making multiple queries and join the data manually within code.

**Memory usage:** MongoDB uses a lot of space as it has to store key names in every document. There is also so much of duplicate data.

**Transactions:** MongoDB doesn’t treat operations as transactions. It should be done manually by choosing to create transaction and verifying it.

**Final conclusion:** Even though the cons list is bigger it doesn’t mean that MongoDB shouldn’t be used, developers shouldn’t use MongoDB as replacement to relational databases it is designed for specific scenario in which it excels. Don't do things just because they're cool; do them because it makes sense. MongoDB is a very powerful tool if used in keeping mind the limitations of it