**COS20015 (Fundamentals of Database Management)**

**Research Report: Is Relational database better than NoSQL database?**

**(using MariaDB and MongoDB to compare and contrast)**

Name: SM Ragib Rezwan

ID: 103172423

CONTENT:

* Abstract
* Introduction
* Method
* Result
* Discussion
* Conclusion
* MongoDB database setup
* Reference

**Abstract:**

Databases are basically ways to store and access groups of data in an organized, efficient manner. Although there are several types of them currently existing in the market, they are all used to perform these two functions, albeit in slightly different ways (depending upon the needs of the company).

Among them, the most popular and well-known one is Relational Database. But NoSQL databases are also rising in demand in this current fast paced world. This arises the question: Is Relational database better than NoSQL database? Or has the time come to switch to NoSQL?

**Introduction:**

The dawn of the age of internet has made the world increasingly interconnected, leading to a high demand in storage, access, modification and deletion of data for both personnel and organizations. This has led to the development of a group of software, each of which can accomplish these tasks in different ways. This group of software is what we know as databases.

Currently in the world, different types of databases (like Centralized, Cloud, Object Oriented databases, etc) exists, each using different Query languages to help the users find and use the information they need effectively, reliably and efficiently. But even so most organizations generally tend to focus on two types: Relational and NoSQL databases.

Relational databases are the type of databases where structured information is stored following a rigid schema. Here, tables are made for each entity and relationships are made to connect those tables (and thus link the entities). Also, each table has columns which denote the attributes associated with that entity and rows to denote the records related to that entity. Furthermore, this type of databases either use SQL or a “dialect” of it (ie MYSQL) to search though its tables for specific data to access, update or delete and follows normalization principles (at least upto 3NF) to reduce data redundancy and increase data integrity.

On the otherhand, NoSQL databases are the types which store all forms of data (with main advantage laying in the use of semi structured and unstructured data), without enforcing any schema on the data stored. Thus, it can be realized that they don’t follow normalization principle at all! Furthermore, although there are several types of such database (like Key-Value storage, Graph database, Wide-column Store, etc.) available, the only difference between them is in the slightly different way the data is stored and everything else is more or less the same.

So, although Relational and NOSQL databases are quite different from one another, we can see that they still basically perform the same functions as every other databases. Hence comes the obvious question: Is Relational database better than NoSQL database? Or has the time come to switch to NoSQL ones?

**Method:**

Although several Relational (like Oracle, PostgreSQL, SQL server, etc.) and NOSQL databases (like HBase, Cassandra, Haystack, etc.) exist, here I am going to only use MariaDB and MongoDB to compare them. This is because:

1. Both of them are very popular and most commonly used in the current world
2. Both of them are open source, enabling me to freely modify and use it without worry
3. Both of them have been used in varying degree in this course. So, I have a good idea on what they can and can’t do, and thus, know most of the external factors that I need to control (to ensure proper comparison can be made between them).

To ensure that the comparison is fair, I am going to use the database I previously made in MariaDB (for my “custom database” task) and utilize its data to make the databases in MongoDB. This ensures that both databases have the same data stored in them and so any differences present will be due to the database type used to store them.

*[Note: if you wish to see the database setup and full details for MariaDB, please see the pdf for the custom database task]*

*[Note: if you wish to see the database setup and full details for MongoDB, please see between conclusion and reference]*

1. **Brief information regarding the database made in MariaDB:**

Here I used MYSQL query language to make a database called companynew which had tables for: Product\_details, subsidiary\_company\_details, shipments, Orders, Customers\_details, Employee\_details, Individual\_Employee\_position, Employee\_position, Employee\_and\_Branch\_details and Branch\_details.

In Product\_details table, there were attributes for product\_id, product\_name, product\_type, product\_material, product\_price, product\_comment, subsidiary\_company\_id, product\_purchase\_date\_from\_subsidiary,product\_in\_stock. The table also had 5 rows of data in it, with primary key product id and foreign key subsidiary\_company\_id.

In subsidiary\_company\_details table, there were attributes for sunsidiary\_company\_id, subsidiary\_company\_name, subsidiary\_company\_address, subsidiary\_company\_city,subsidiary\_company\_postcode, subsidiary\_company\_phonenumber. The table also had 4 rows of data in it with primary key subsidiary\_company\_id

In shipment table, there were attributes for shipment\_id, order\_id,shipment\_date . The table also had 4 rows of data in it, with primary key shipment\_id and foreign key order\_id.

In orders table, there were attributes for order\_id, customer\_id, product\_id, employee\_id, order\_stock, order\_date, order\_comment. The table also had 4 rows of data in it with primary key order\_id and foreign key customer\_id, product\_id and employee\_id. Also, it is a weak entity table

In customers\_details table, there were attributes for customer\_id, customer\_fname, customer\_lname, customer\_gender, customer\_address, customer\_city, customer\_postcode, customer\_phone . The table also had rows of data in it with primary key customer\_id.

In Employee\_details table, there were attributes for mployee\_id, employee\_fname, employee\_lname, employee\_gender, employee\_date\_of\_birth, employee\_address, employee\_city, employee\_postcode, employee\_phone, employee\_current\_salary, employee\_individual\_comment. The table also had 5 rows of data in it with primary key employee\_id.

In Individual\_Employee\_position table, there were attributes for employee\_id, employee\_position . The table also had 5 rows of data in it with foreign key employee\_id, employee\_position. Also, it is a weak table.

In Employee\_position table, there were attributes for employee\_position and employee\_position\_comment . The table also had 3 rows of data in it with primary key employee\_position.

In Employee\_and\_Branch\_details table, there were attributes for branch\_id, employee\_id . The table also had 5 rows of data in it with foreign key branch\_id, employee\_id. Also, it is a weak table

In Branch\_details table, there were attributes for branch\_id, branch\_name, branch\_comment . The table also had 2 rows of data in it with primary branch\_id.

Furthermore I have also made 3 views in it for full\_employee\_details, full\_order\_details and full\_product\_details with necessary attributes (via de-normalization) to enable users to easily find all important the data in just 3 places, instead of searching through all the tables present in the database and wasting time.

1. **Brief information regarding the database made in MongoDB:**

[Note: MongoDB is a document database which is basically a subclass of the key-value store database of NoSQL and follows JSON format]

Here I used NOSQL to make a database called newcompanydb and a collection in it called newCompany. Inside that collection, I made 3 documents for branch, subsidiary company and customer details.

In Branch, I made attribute/keys: ID, name, comment for branches and an array called Employee which contained ID, name, gender, date of birth, address (another array inside it that had address, city and postcode), phone, salary, position, comment for all employees in that branch.

In Subsidiary Company, I made attribute/keys: ID, name, city, phone for subsidiary companies and an array called Products which contained ID, name, type, material, price, comment, purchase\_date, stock for all products supplied by that subsidiary company.

In customer details, I made attribute/keys: ID, name, gender, address, city, postcode, phone for customers and an array called orders which contained order ID, product ID, Product name, Ordered Product stock, order product cost, employee ID (of one who sold the product), employee name(of one who sold the product) , order\_date, shipment\_ID, shipdate, Order comment for all orders made by those customers

Here, I am going to compare them on the following matrixes: Data structures and retrieval, concurrency model/isolation levels, speed of access and database security. For each of them I will do brief literary comparison with code implementations to back them up in results. Then I will provide further details and analysis about each in the discussions.

**Results:**

1. **Data structures and retrieval:**

*[Note: Data structure” is the way used to provide an efficient method of storing and organizing groups of data elements in “temporary memory”, so that they can be easily modified and accessed. This is different from “database structure” which manages data stored in “permanent memory]*

In the MariaDB, MySQL provides good data structure and retrieval by utilizing techniques like denormalization via views and by using indexes (like primary, B+ Tree, and Hash).

Command and output for a sample view:

Create View Full\_Employee\_Details as

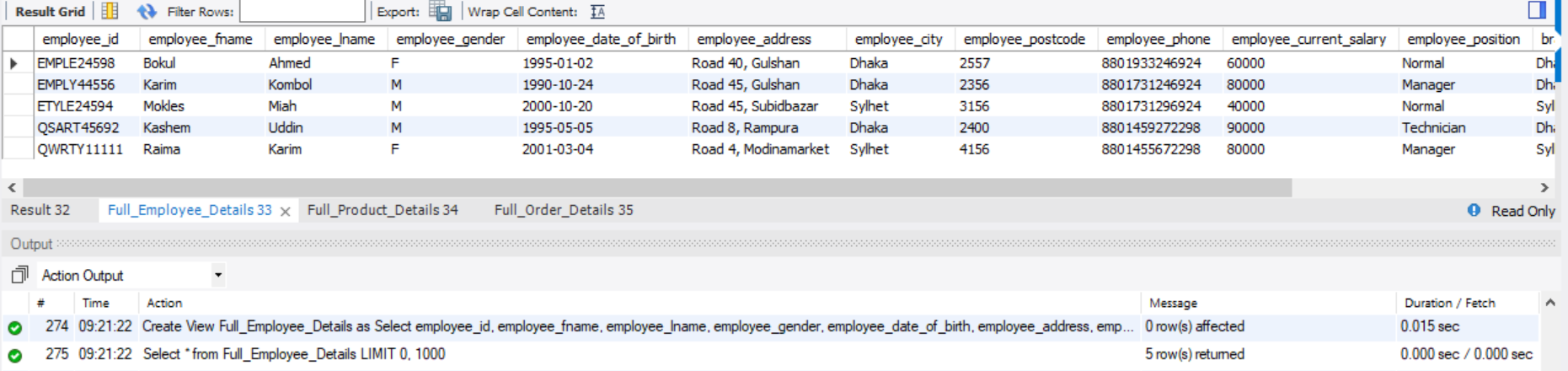
Select employee\_id, employee\_fname, employee\_lname, employee\_gender, employee\_date\_of\_birth, employee\_address,employee\_city, employee\_postcode, employee\_phone, employee\_current\_salary, employee\_position,branch\_name, employee\_individual\_comment

From employee\_details e Natural Join employee\_and\_branch\_details eb Natural Join branch\_details b NATURAL Join individual\_employee\_position ie NATURAL Join employee\_position

Order by employee\_id;

Select \*

from Full\_Employee\_Details;



On the other hand, in MongoDB, all the objects and attributes related to the entity are kept in a single document. So, only need to query that document normally to retrieve the data, instead of implementing any extra techniques.

db.newCompany.find().pretty()

Command used:



Output:

1. **concurrency model/isolation levels:**

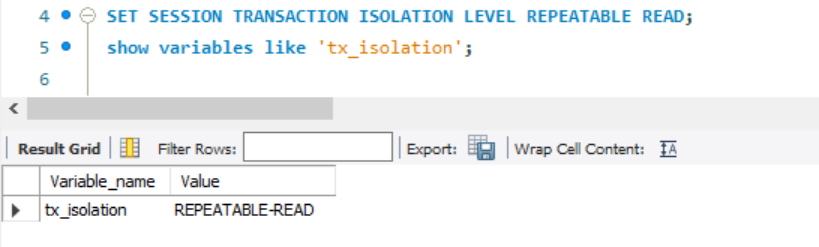
*[Note: Data Concurrency is the ability to let many users work on the same data at the same time, (keeping it upto date) which increases the risks of errors (like Lost update, Dirty reads, non-repeatable reads, phantoms, etc.) when proper precautions are not maintained]*

In MySQL, this is maintained by using isolation levels: Read Uncommitted, Read Committed, Repeatable Read, and Serializable; with the default being REPEATABLE READ. This is usually altered by using “SET SESSION TRANSACTION ISOLATION LEVEL <<Isolation level name>>” Command before performing the transaction.

Command used:

SET SESSION TRANSACTION ISOLATION LEVEL REPEATABLE READ //setting to repeatable read

Show variable like ‘tx\_isolation’ //wrote this so that the change can be observed

Output:

In NOSQL, it is maintained by using ReadConcern and Write concern, where read concern can be set as local, available, majority, linearizable and snapshot (with default being local) and write concern can be set as majority, 0, 1, 2 (or any other number), or as a custom name (with default being majority). The read conditions for transaction can be set by adding readconcern and level after the desired query ( eg “db.collection.find().readConcern(<level>)”command) and write concern can be set using the “db.setWriteConcern(<level>)” command

*[Note: For defaults for read and write concern, I had to look up on website online, as in the MongoDB I used, there weren’t any default values given for them from before. The website I used has been referred to in the discussion part for this area]*

Command used:

//to show name of database used

//to set the value to 1 which means it will acknowledge from primary before performing the task

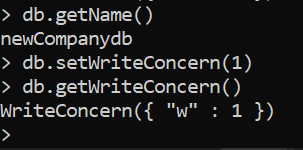
//to show that it has been set

db.getName()

db.setWriteConcern(1)

db.getWriteConcern()

Output:



1. **Speed of access :**

In order to measure speed of access, I ran a query on both databases and noted the time it took for them to run it. Since I want both queries output the same result, I tried to find “all employee details, including their branch details” using the query.

*[Note: set profile and show profile are used here to track and measure time taken to run the query]*

For MariaDB, it can be done in two ways:

1. commands used are (without using technique like view):

set profiling=1;

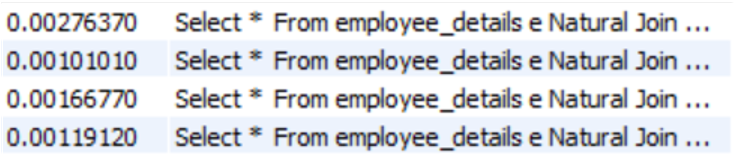
Select \*

From employee\_details e Natural Join employee\_and\_branch\_details eb Natural Join branch\_details b

Order by employee\_id;

show profiles;

And the times it took:

So, on average it took: **0.00165818 second**

**(1.65818 milliseconds)**

1. commands used are ( using view):

set profiling=1

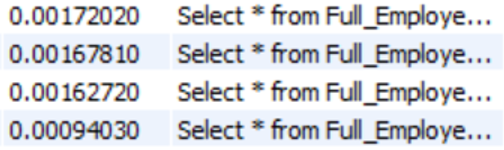
Select \*

from Full\_Employee\_Details

Order by employee\_id;

show profiles;

And the time it took:

So, on average it took:

**0.00149145 second**

**(1. 49145 milliseconds)**

For MongoDB, commands used are:

db.setProfilingLevel(2)

db.newCompany.find( { "\_id" : ObjectId("6177c86fd5d09d331b8a6198") }).pretty().explain("executionStats")

And the output is:



This output stated that the time taken was 0 milliseconds. This seemed unusual to me and thus I ran it several times, while controlling all external factors (ie keeping all other applications off, checking just after laptop had boot up, etc). Even so, it still kept coming 0 milliseconds. So had to consider this data as accurate, noting the time as either 0ms, or very, very close to that.

1. **Database security:**

*[Note: Database Security is the use tools or methods in a database to ensure security of the data in it]*

In MariaBD, it is done by End-to-end encryption, by Firewall and data masking, by Auditing, by Denial of service protection and by Pluggable authentication and role/group authorization. Each of these performs in different ways, providing protection against various concerns in the database. For instance, End to end encryption is done using AES and TLS where AES is used to encrypt the data being inserted to a table.

//here it encrypts the string “Bigar” inserted into the table using the key key\_string “SHA2” and stores a binary string in the field.

Use test;

create table t

(name\_stuff blob not null);

INSERT INTO t (name\_stuff) VALUES (AES\_ENCRYPT('Bigar',SHA2('password',512)));

SELECT name\_stuff,

AES\_DECRYPT(name\_stuff, SHA2('password',512) )

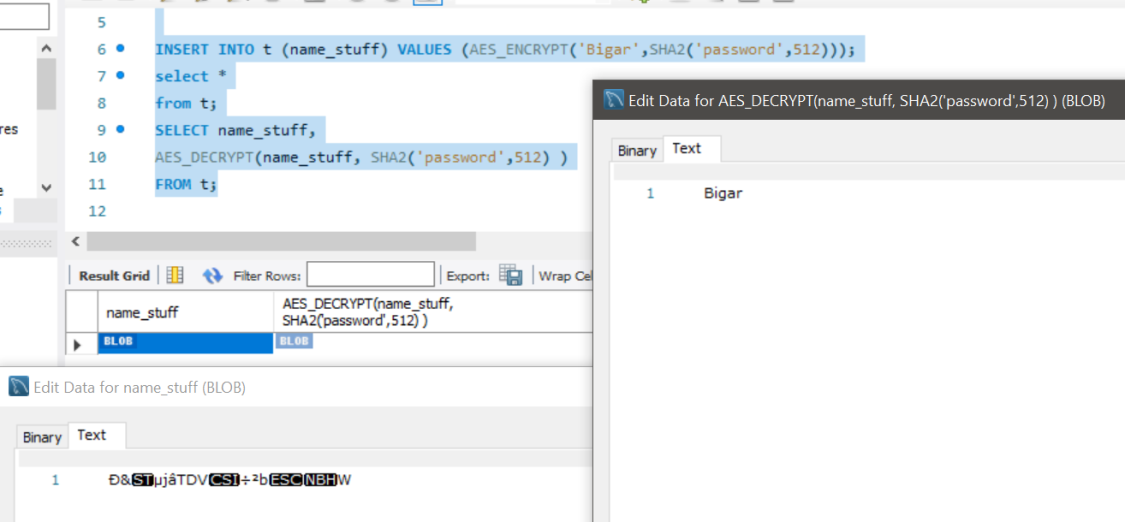
FROM t;

//Furthermore, the only condition here is that the datatype of the field must be BLOB.

//Here I wrote two columns in the query, one to show how it looked encrypted and one to show what the actual data was

*[In practical case, just directly use AES\_DECRYPT part only and ignore name\_stuff part]*

*[Note: If you want to use VARCHAR, rather than BLOB, then convert the encrypted binary to Base64]*

 Outputs:

On the other hand, in MongoDB, it is done by encryption, authentication, authorization, auditing. Like in case of MariaDB, each these focus on providing protection against different threats to the database. For instance, for encryption MongoDB uses Encrypted storage engine which ensures data is encrypted both on network and on disk. It can be set using the following command:

|  |
| --- |
| clientEncryption = db.getMongo().getClientEncryption(); |
| "phone" : clientEncryption.encrypt( |
| UUID("64e2d87d-f168-493c-bbdf-a394535a2cb9"), |
| "123-45-6789", |
| |  | | --- | | "AEAD\_AES\_256\_CBC\_HMAC\_SHA\_512-Deterministic") | |
|  |
|  |

//getting the Client encryption, storing it in variable, then encrypt by using a key (the UUID), value you want to encrypt (ie the phone number) and an algorithm you wish to use to encrypt it.

*[Note: Unfortunately, this can only be used in mongodb’s enterprise mode (not in the community mode). Thus I cant show the output for it. But basically you will have to just set up a strict schema saying what sort of encryption the objects will have and then run the commands given to insert the data using the encryption]*

**Discussion:**

* In data structures and retrieval, MariaDB has to use techniques like view and indexing to maintain it. Although denormalizing using views sounds counter intuitive with respect to its “relational database structure”, using views actually helps saves time and resources for big queries. That’s because it creates a pre-optimized table that contains data of multiple tables stored in it, avoiding the use of the expensive joins to query and update data. Furthermore, with the help of indexing (which is basically a look up table), pointers can be assigned to values on one or more columns (like on only primary key for primary index, any attribute for B+ tree, attributes that are highly selective for Hash maps) making it more efficient for databases as they only have to see the starting row at each block to find where their desired data is located (instead of checking all the rows in all blocks).

On the other hand, in MongoDB’s case, it is a document database where related entities are stored in a collection using JSON format (attribute stored as name-value pair separated by colon, different attributes separated by commas, and objects/sub-documents related to that entity) separated using curly brackets). Thus, (as seen in the results) we don’t have to use any extra techniques (like creating and maintaining views or anything else) as everything related to an entity are kept in a single document, instead of being separated into multiple tables. So only need to query that to retrieve the data, which is very efficient and also more readable (as the name beside the value directly tells you what it represents).

Thus, regarding to data structure and retrieval, it is better to use MongoDB instead of MariaDB as you can find all the needed data in one place (without using any extra techniques) in a readable manner from the getgo, instead of looking through and linking multiple related tables.

* In terms of Concurrency, MariaDB does this by maintaining isolation levels, Read Uncommitted, Read Committed, Repeatable Read, and Serializable, with read uncommitted having highest performance and serializable having the highest isolation level. Read committed stops no anomaly what so ever and is the worst one to be used here. Next level is Read Committed which stops lost update (update done to data in a transaction lost as it was overwritten by another at the same time) and dirty reads (transaction reading a data that has been modified but not committed by another transaction). After this comes Repeatable Read which stops lost update, dirty read and also non-repeatable read (transaction reads a data without committing but finds different value when it re-reads the same data). At the very last comes Serializable which basically stops all anomalies (including phantoms where rerunning the same query turns up new rows) but needs to wait for each transaction to run one at a time, which makes it slow.

In MongoDB, it is maintained a bit more flexibly, allowing people to alter read concern and write concern to set different isolation levels of reading and writing the data. According to the MONGODB Manual, by default, read concern is set as local (query instantly returns data without guarantee whether majority of replica set received it or not, available for use with or without causally consistent sessions and transactions) and write concerns is set as majority (need acknowledgement from majority of the primary and secondary present before writing to it).

*[Note:**Causal consistency means Operation that logically depends on a preceding operation]*

Furthermore, for read concern, they can also be set as available (same as local, but only used without causally consistent session and transaction), majority (returns data that majority of replica set acknowledged), linearizable (returns data that majority-acknowledged write has successfully completed, only used without causally consistent session and transaction) and snapshot (read data after returned from transaction complete with write concern majority).

Moreover, for write concern, they can also be set as 0, 1, 2 or any other number (depends on how many primary and secondary Requests must be acknowledged for the write operation) or as custom name (needs acknowledgement from tagged members before writing).

Although, we can see MongoDB has good flexibility and control in terms of reading and writing data, it doesn’t actually support transactions. Instead, it offers transaction-like semantics. So, it is possible for application to return an intermediate data during commit or rollback, which can lead to dirty read scenario. Thus, in this case, using MariaDB is better than using MongoDB.

* In terms of speed of access, we can see that although both databases are very fast (as they take milliseconds on average to run the queries outputting the same data for each), MongoDB runs the query the fastest (as it takes 0ms or a time very close to it on average). That’s mainly because in MongoDB, it just needs to load up a single document while in MariaDB, it needs to either join multiple tables or run a pre-optimized view. Thus, in terms of speed, MongoDB is far, far ahead of MariaDB and thus for high speed performance, it is better to use MongoDB rather than MariaDB.
* In terms of database security, MariaDB does it in 5 ways: End-to-end encryption, Firewall and data masking, Auditing, Denial of service protection and Pluggable authentication and role/group authorization. End-to-end encryption encrypts the data passed using AES(Advanced Encryption Standard) and also the connection itself using TLS(Transport Layer Security) (as seen in the commands before). Firewall and data masking is used to prevent malicious attacks like SQL injection and to hide sensitive data by intercepting and blocking certain queries (implemented by making a masking filter using Maxscale configuration and a json file with masking rules detailed in it). Auditing tracks everything on the database ( all queries, tables operations, etc) including when and who made the change (need to install Mariadb audit plugin, activate it and choose which events to audit like connection, queries, tables, etc). Denial of service protection mainly prevents Dos and DDos attacks (implemented by configuring a result limiting filter to block queries that intend to slow down database by returning huge results). Pluggable authentication and role/group authorization ensure that only authenticated and authorized personnel can access and change the data in the database (implemented using LDAP authentication, one-time passwords, two factor authentication, setting up user accounts and its password,etc)
* On the other hand, in MongoDB, it is done by 4 ways: Encryption, Authentication, Authorization, and Auditing. Encryption is done using MongoDB’s Encrypted storage engine which ensures data is encrypted both on network and on disk. Authentication is done by integration LDAP, Kerberos, etc external security to ensure person has the needed authentication to access the data. Authorization is done by providing different roles to user for the application, limiting their access to only that which is needed. Auditing is done by using a native log that keeps tracks of all operations and access performed on the database.

*[Note: I have been unable to set the encryption on MongoDB as I don’t have access to MongoDB’s Enterprise Mode and thus stated what would be seen and the way to implement it, following the MongoDB manual]*

Now, although these two have similar security, there is one feature that separated both of them. In MariaDB, there are features present to prevent DDOS attacks which is absent in MongoDB. Thus considering the current situation, where DDoS attack has been on the rise (as seen by David Warburton in F5 labs where he noticed that it has increased by 55% between Jan 2020 to March 2021), MariaDB has an easy win compared to MongoDB.

**Conclusion:**

So overall, when we compare MariaDB and MongoDB, we can see that they both have different benefits and drawbacks. This, in turn, leads them to be to be highly effective in some situations and poor in others. For instance, if a company used only structured data, demands good transaction and database consistency, no redundancy, robust database, high database security, etc. then they should use MariaDB as it is well suited for that situation. But if they need to deal with unstructured or semi-structured data, have fast access from getgo (without relying on extra techniques), flexible database schema, etc. then they should use MongoDB instead.

But does this also apply for all relational and NoSQl database? That is not certain.

That’s because:

1. Here we only compared among two databases (MariaDB and MongoDB) of those two types (Relational and NoSQL databases) instead of all databases that follow these two types.
2. Also, when we compared MariaDB and MongoDB, only data allocated to one company’s (ie my imaginary “furnitures international” company) use was noted, which lead to a certain format for MongoDB’s setup (ie 3 documents containing everything important). But different formats (like making individual documents for each person or product, keeping data in different collections in the database, etc) could have also been written for it. And, since not all possible formats have been tested, there isn’t a 100% guarantee that the comparisons written here will also be accurate for those cases as well.
3. Furthermore, as seen before, the encryption command need MongoDB’s enterprise version to run the command. This showed that MongoDB’s Enterprise had more commands, which were unavailable to those accessing it using MongoDB’s Community version. Thus, it is possible that there were other commands present there which may had given MongoDB a better advantage against MariaDB if used here.

So, It will be better to repeat this experiment firstly on different ways the database could also be made in MongoDB Enterprise (to have full access to all commands) and then on other Relational (like Oracle, PostgreSQL, SQL server, etc.) and NOSQL databases (like HBase, Cassandra, Haystack, etc.) that also exists. It is only then that we can make an accurate, valid, clear-cut answer to the question:

“Is Relational databases better than NoSQL databases?”

**MongoDB database setup:**

use newcompanydb

db.newCompany.insertMany(

[

{"Branch":

[

{

"ID" :"BRNCH40002",

"Name":"Dhaka Branch",

"Comment":"Exclusive Branch in Dhaka",

"Employee":

[

{

"ID": "EMPLE24598",

"Name": "Bokul Ahmed",

"Gender": "F",

"Date\_of\_birth" : "1995-01-02",

"Address":

{

"address": "Road 40, Gulshan",

"city":"Dhaka",

"postcode":"2557"

},

"Phone":"8801933246924",

"Salary":"60000",

"Position":"Normal",

"Comment":"Clumsy Employee"

},

{

"ID": "EMPLY44556",

"Name": "Karim Kombol",

"Gender": "M",

"Date\_of\_birth" : "1990-10-24",

"Address":

{

"address": "Road 45, Gulshan",

"city":"Dhaka",

"postcode":"2356"

},

"Phone":"8801731246924",

"Salary":"80000",

"Position":"Manager",

"Comment":"good old employee"

},

{

"ID": "QSART45692",

"Name": "Kashem Uddin",

"Gender": "M",

"Date\_of\_birth" : "1995-05-05",

"Address":

{

"address": "Road 8, Rampura",

"city":"Dhaka",

"postcode":"2400"

},

"Phone":"8801459272298",

"Salary":"90000",

"Position":"Technician",

"Comment":"old techy employee"

}

]

},

{

"ID" :"BRNCH20003",

"Name":"Sylhet Branch",

"Comment":"Exclusive Branch in Sylhet",

"Employee":

[{

"ID": "ETYLE24594",

"Name": "Mokles Miah",

"Gender": "M",

"Date\_of\_birth" : "2000-10-20",

"Address":

{

"address": "Road 45, Subidbazar",

"city":"Sylhet",

"postcode":"3156"

},

"Phone":"8801731296924",

"Salary":"40000",

"Position":"Normal",

"Comment":"Lazy employee"

},

{

"ID": "QWRTY11111",

"Name": "Raima Karim",

"Gender": "F",

"Date\_of\_birth" : "2001-03-04",

"Address":

{

"address": "Road 4, Modinamarket",

"city":"Sylhet",

"postcode":"4156"

},

"Phone":"8801455672298",

"Salary":"80000",

"Position":"Manager",

"Comment":"Diligent employee"

}]

}

]

},

{

"Subsidiary Company":

[

{

"ID": "SCOMP00001",

"Name":"Office Products",

"City":"Dhaka",

"Phone":"8801972448031",

"Product":

[{

"ID": "PHGTA00144",

"Name": "Office table",

"Type": "table",

"Material": "wood",

"Price": "6000",

"Comment": "good office table made of wood",

"Purchase\_Date":"2019-02-16",

"Stock":"40"

},

{

"ID": "PRDTA00103",

"Name": "Office chair",

"Type": "chair",

"Material": "wood",

"Price": "4000",

"Comment": "good office chair made of wood",

"Purchase\_Date":"2020-12-12",

"Stock":"70"

}]

},

{

"ID": "SCOMP00051",

"Name":"RFL",

"City":"Dhaka",

"Phone":"8801745336711",

"Product":

{

"ID": "PRDTA00140",

"Name": "RFL sofa",

"Type": "sofa",

"Material": "plastic",

"Price": "500",

"Comment": "good RFL chair made of plastic",

"Purchase\_Date":"2018-10-12",

"Stock":"90"

}

},

{

"ID": "SCOMP00444",

"Name":"Stylish Products",

"City":"Dhaka",

"Phone":"8801973448031",

"Product":

{

"ID": "PRDTG00210",

"Name": "stylish chair",

"Type": "chair",

"Material": "wood",

"Price": "10000",

"Comment": "good chair made of wood ",

"Purchase\_Date":"2020-12-15",

"Stock":"50"

}

},

{

"ID": "SCOMP05001",

"Name":"Glassify",

"City":"Sylhet",

"Phone":"8801745336700",

"Product":

{

"ID": "PROOA00103",

"Name": "clear wardrobe",

"Type": "wardrobe",

"Material": "glass",

"Price": "8000",

"Comment": "good wardrobe made of glass ",

"Purchase\_Date":"2020-10-10",

"Stock":"90"

}

}

]

},

{

"Customer Details":

[

{

"ID": "ADBCA00554",

"Name": "Kawsar Hossain",

"Gender": "M",

"Address": "Road 10, Bonani",

"City": "Dhaka",

"Postcode": "4156",

"Phone":"8801796958971"

},

{

"ID": "ALBCA11020",

"Name": "Anne Stuward",

"Gender": "F",

"Address": "Road 2, Rampura",

"City": "Dhaka",

"Postcode": "6915",

"Phone":"8801692433190"

},

{

"ID": "FABCA09548",

"Name": "Robert Brown",

"Gender": "M",

"Address": "Road 10, Modina",

"City": "Sylhet",

"Postcode": "3114",

"Phone":"8801892884308",

"Order":

{

"Order ID" :"ORDRE44990",

"Product ID" :"PROOA00103",

"Product name" :"clear wardrobe",

"Ordered product stock" : "10",

"Order product cost" : "80000",

"Employee ID" : "ETYLE24594",

"Employee name" :"Mokles Miah",

"Order date" : "2021-06-26",

"Shipment\_ID":"SHPIN40705",

"Shipdate" : "2021-06-28",

"Order comment" :"good old customer back"

}

},

{

"ID": "GABCA07324",

"Name": "Salma Begum",

"Gender": "F",

"Address": "Road 21, Gulshan",

"City": "Dhaka",

"Postcode": "2114",

"Phone":"8801592457044",

"Order":

{

"Order ID" :"ORDRE49100",

"Product ID" :"PRDTA00103",

"Product name" :"Office chair",

"Ordered product stock" : "10",

"Order product cost" : "40000",

"Employee ID" : "EMPLE24598",

"Employee name" :"Bokul Ahmed",

"Order date" : "2021-07-07",

"Shipment\_ID":"SHPIN55555",

"Shipdate" : "2021-07-09",

"Order comment" :"customer wants to track this"

}

},

{

"ID": "HABCA00024",

"Name": "Sultan Ahmed",

"Gender": "M",

"Address": "Road 2, Gulshana",

"City": "Dhaka",

"Postcode": "2141",

"Phone":"8801792458030",

"Order":

[

{

"Order ID" :"ORDRE45679",

"Product ID" :"PHGTA00144",

"Product name" :"Office table",

"Ordered product stock" : "1",

"Order product cost" : "6000",

"Employee ID" : "EMPLE24598",

"Employee name" :"Bokul Ahmed",

"Order date" : "2021-05-24",

"Shipment\_ID":"SHPIN30078",

"Shipdate" : "2021-05-26",

"Order comment" :"customer felt weird"

},

{

"Order ID" :"ORDRE45680",

"Product ID" :"PRDTA00103",

"Product name" :"Office chair",

"Ordered product stock" : "1",

"Order product cost" : "4000",

"Employee ID" : "EMPLE24598",

"Employee name" :"Bokul Ahmed",

"Order date" : "2021-05-24",

"Shipment\_ID":"SHPIN30079",

"Shipdate" : "2021-05-26",

"Order comment" :"customer felt weird"

}

]

}

]

}

]

)

**Reference:**

<https://www.javatpoint.com/types-of-databases>

<https://pediaa.com/what-is-the-difference-between-database-and-data-structure/>

<https://mariadb.com/kb/en/storage-engine-index-types/>

<https://www.infoq.com/articles/Isolation-Levels/>

<https://docs.mongodb.com/manual/core/read-isolation-consistency-recency/>

<https://www.mongodb.com/nosql-explained>

<https://www.mongodb.com/developer/how-to/global-read-write-concerns/>

<https://docs.mongodb.com/manual/reference/write-concern/>

<https://docs.huihoo.com/mongodb/3.4/tutorial/perform-two-phase-commits/index.html>

<https://mariadb.com/database-topics/security/>

<https://mariadb.com/kb/en/mariadb-maxscale-21-masking/>

<https://www.mongodb.com/scale/nosql-database-security>

<https://www.f5.com/labs/articles/threat-intelligence/ddos-attack-trends-for-2020>

<https://mariadb.com/kb/en/mariadb-maxscale-21-masking/>

<https://mariadb.com/kb/en/show-profile/>

<https://mariadb.com/kb/en/mariadb-audit-plugin/>

<https://mariadb.com/kb/en/authentication-from-mariadb-104/>