**MySQL vs MongoDB**

A comparison of MySQL and MongoDB involves understanding their underlying technologies – relational (SQL) and non-relational (NoSQL) databases. Relational databases require structured tables with a set schema that defines the relationships between tables. Related information may be stored in different tables, thereby reducing data duplication. SQL database sizes are limited by the server storage capacity available and need vertical scaling to store additional data. SQL databases emphasize the ACID (Atomicity, Consistency, Isolation, Durability) approach to transactions to ensure reliability.

The rise of Big Data resulted in a shift towards NoSQL databases. Big Data, which is characterized by volume, variety, and velocity, requires databases that permit unstructured data and frequent updates. NoSQL databases do not enforce a specific schema, and documents can be created without initially defining the database structure. They also permit a combination of structured, semi-structured and unstructured data in the same database. NoSQL databases are not limited by server storage capacity since they are horizontally scalable by adding more servers. NoSQL databases emphasize the CAP (Consistency, Availability, Partition tolerance) approach to data, prioritizing availability over data integrity.

Different types of NoSQL databases exist, chief among which is document-oriented, column-oriented, key-value stores, and graph DB’s. MongoDB is a NoSQL database that is document-oriented, that is, a document is the basic storage unit and all data is stored in one document which is treated as a single object. This document is formatted in BSON, an extension of JSON. Different data types can be stored and updated in a single document. It is also possible to embed a document store inside another to facilitate multi-table querying. MongoDB can handle rapid data growth better than MySQL through automatic sharding across multiple servers, even in different locations. Using sharding, MongoDB enables the distributed storage of data across multiple servers, thus accommodating for the data growth needs of users. Querying across data shards is accomplished by the query router, which manages the allocation of queries to the relevant shards and return of the result.

**Concepts/Terminology:**

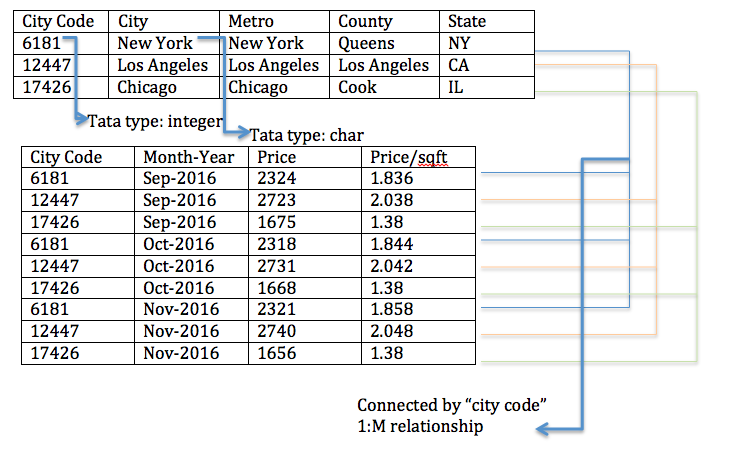
|  |  |
| --- | --- |
| **MySQL** | **MongoDB** |
| Database | Database |
| Table | Collection |
| Row | Document (JSON) |
| Index | Index |
| Primary key is an integer unique for each row | Primary key is a BSON Object ID |
| Join | $lookup, embedded documents |
| Vertically/Scale-up | Horizontally/Scale out |
| Integrity constraints such as type, value, range | No constraints on updating data |

**Framework difference:**

|  |  |  |
| --- | --- | --- |
|  | **MySQL** | **MongoDB** |
| Coding | C++, C | C++, C, JavaScript |
| Type | RDBMS | Document |
| Created by | MySQL AB | MongoDB.Inc |
| Open-Source | YES | YES |
| Owner | Oracle | MongoDB, Inc |
| Schema | Strict | Dynamic |

A major reason for the difference between MySQL and MongoDB in query preparation and execution is how data is stored.

**MySQL:**

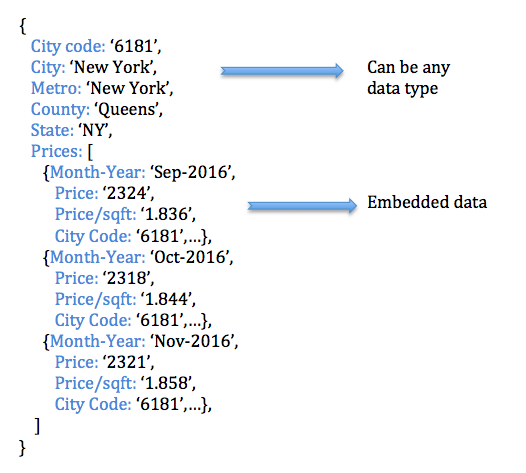


MySQL requires data to be stored in tables, with all rows in the tables having the same data type. For e.g. in the table above, “City Code” has data type “integer” and “City” has the data type “char”. Relationships between tables are defined by 1:1, 1: M, and M: M linkages. The linkages among tables facilitate multi-table querying through the use of JOIN statements. The limitations on variability in data types mean that data cleaning and normalization are necessary when preparing the data for queries.

Querying is done using the Structured Query Language for tabular data and is each part of the query is executed sequentially. Therefore, the order of queries (query syntax) is important. Indexes are used to speed up the performance of queries.

Processing of an SQL statement consists of various steps: Parsing (checking for syntax and semantics), Optimization and Query Plan Generation (determining the most efficient way to execute the query from multiple alternatives including cached query plans), and Execution Plan (to run the query and return the results as a result set).

**MongoDB:**



Data is stored in documents, and related documents are stored as collections. The fields in a document can have any data type, which results in faster updates when compared to MySQL. Related documents can be embedded, thus removing the need for defining linkages and reducing the time needed for data cleaning.

Querying is done using Javascript that can be applied to unstructured data. In case of huge databases, MongoDB allows for cluster querying where queries are allocated to specific shards which house the relevant data and return the final result. Use of indexes in MongoDB is similar to MySQL; however, MongoDB automatically generates a default index while in MySQL, the index has to be predefined through primary keys.

The MongoDB query optimizer processes the query and chooses the most efficient query plan for a query given the available indexes. The query system then uses this query plan each time the query runs.

**Query Syntax:**

|  |  |  |
| --- | --- | --- |
| **CRUD Operations** | **MySQL** | **MongoDB** |
| Create | INSERT INTO table\_name  (column1, column2, column3...) VALUES (“value1”, “value2”, integer3...); | db\_name.collection\_name.insert({  field1 : “value1”,  field2 : “value2”,  field3 : integer3}) |
| Read | SELECT column\_name(s) FROM table\_name WHERE condition; | db\_name.collection\_name.find({ field\_name : <condition>},  { field\_name : 1, ..}) |
| Update | UPDATE table\_name  SET column1 = value1, column2 = value2, ...  WHERE condition; | db\_name.collection\_name.updateMany(  { field\_name : <condition>},  { field\_name : <new value>}) |
| Delete | DELETE FROM table\_name  WHERE condition; | db\_name.collection\_name.deleteMany({field\_name : <condition>}) |

**Performance, Data Integrity, and Security Evaluation:**

MongoDB is designed for Big Data storage and querying, focusing on social network applications. MongoDB obtains its performance mainly by key-value based design and is easy to scale out. Being a member of the NoSQL databases family, MongoDB uses the document as a basic storage unit. A document is a simple object like JSON.

**Performance:** Due to schema-free design, MongoDB (and other NoSQL databases) are claimed to be faster than SQL. In addition, having related data as embedded or linked documents is another reason for MongoDB to provide faster querying. MongoDB’s usage of BSON to store data make it easier to store data, which in turn leads to better performance when running queries. Using distributed storage and computing, MongoDB provides high performance for large datasets also. That said, MySQL is competitive in terms of performance when it comes to structured data, complex queries, and multi-row transactions. It must be noted that MongoDB’s performance advantages over MySQL come at the expense of features such as ensuring data integrity.

**Data Integrity:** Data Integrity is one of the primary characteristics of a good database/DBMS design. It indicates the accuracy, consistency, and reliability of the data in a database. In a typical RDBMS, data integrity can be enforced while designing the database and/or on the implementation side of application logic.

* MySQL Server will enforce data integrity by utilizing a combination of its inherent features like row-level constraints, column-level constraints, foreign key constraints (referential integrity), check constraints. Some user-defined constraints can also be created while programming the application logic if it is not possible with the constraints inherent to the MySQL server.
* The row-level constraint in MySQL is a primary key constraint which enforces each row in a table to have a unique identifier. It is mandatory to assign a primary key while creating a table in MySQL.
* MongoDB automatically creates an auto-incrementing field called ‘\_id’ to every document in a collection, which is like the primary key in the RDBMS. This avoids the problem of duplicate data.
* Some column-level constraints can be added in MySQL in terms of adherence to data types, length, uniqueness, default values, the range of allowable values and allowance of non-null values. Whenever any of these conditions are not met while inserting data into a table, MySQL throws an appropriate error won’t add the record(s) to the table.
* MongoDB can add validations at the document level per-collection basis and at different levels (Strict, moderate etc.) and can impose uniqueness restriction on fields using a unique index. But MongoDB is not as powerful as MySQL in imposing these restrictions because MongoDB is schema-free.
* MySQL maintains referential integrity based on foreign key constraints. This makes sure that all referenced data is consistent, and no bad data can be inserted for which reference records are not present. This nature contributes to the “consistency” part of the data integrity. As MySQL transactions obey ACID (Atomicity, Consistency, Isolation, Durability) properties, consistency is maintained at all levels of data.
* Whereas MongoDB won’t have any foreign key references, it instead uses embedded documents. Also, MongoDB operations are ACID compliant at the document level. So, consistency can be maintained in case of embedded documents. But MongoDB cannot be ACID-compliant in case of multiple document operations. We need to code the logic to maintain this from the application side.

**What is ACID:** ACID stands for Atomicity, Consistency, Isolation, and Durability.

1. **Atomicity** means if a transaction involves multiple steps like INSERT then UPDATE then SELECT etc., either all the steps should be successfully completed or even if any one step fails all the prior steps should be rolled back. Using BEGIN TRANSACTION and COMMIT TRANSACTION statements, this can be enforced in MySQL. These failures can be anything from data issues to the external power failures.
2. **Consistency** implies that any change made on the database should follow all the pre-defined rules like constraints on keys, triggers, referential constraints etc. If any one rule fails, the database should be left with the previous (Prior to the transaction) state
3. **Isolation** allows multiple concurrent transactions to happen without any issue. If multiple transactions are started at a time, the DBMS processes one transaction after completion of another transaction. This way, during a transaction it isolates all the related objects to keep away from impact from other transactions. This avoids the loss of data or creation of bad data.
4. **Durability** speaks about the data durability once it is stored in the database. Means as soon as the data is stored it should be permanently available unless explicitly dropped or deleted. Even in case of external catastrophes, the data should be retained by means of backup servers or other methods.

**Security:**

* **Authentication & Authorization:** The most basic security that both MySQL and MongoDB provide is authentication and authorization. They allow to create user profiles and give either read or write or both permissions to each user.
* **SQL Injection:** Though MySQL is vulnerable to SQL Injection attacks, this can be avoided by following best coding processes. Like storing user’s input in a variable before parsing it. On the other hand, MongoDB cannot be exploited by SQL injection, because MongoDB stores data in BSON objects, not strings. So, there is no possibility of SQL Injection.
* **Encryption:** MySQL provides various types of encryption to store confidential data. MongoDB also provides encryption for at-rest and in-transit data. Thus, both the technologies secure the data from being extracted by unauthenticated parties.
* **Auditing:** Both MySQL and MongoDB provide auditing capabilities and tools to audit the entire database system. This allows database administrators to track changes and debug errors.
* **Firewall:** MySQL offers its own firewall whereas MongoDB needs to be configured with third-party firewalls. Both technologies are compatible with most of the common firewalls available. Basically, a firewall blocks unwanted network traffic from accessing the data.

**Usage Preference of MySQL and MongoDB:**

MySQL is preferred where there is strong dependence on multi-row transactions and high level of data integrity is required, such as banking systems, accounting, and flight reservations.

MongoDB is used in situations where high availability is preferred, and real-time data needs to be stored, such as location-based data, social media, and data generated by the internet of things sensors. Such data is created rapidly and in large volumes, and it is typically difficult to enforce a uniform type of structure on it.

**Sources:**

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

<https://www.mongodb.com/blog/post/thinking-documents-part-2>

<http://www.geeksengine.com/database/design/data-integrity.php>

<https://www.infoworld.com/article/3164504/security/the-essential-guide-to-mongodb-security.html>

<https://www.upguard.com/articles/mysql-vs-mongodb>

<https://docs.oracle.com/database/121/TGSQL/tgsql_sqlproc.htm>