### What do you understand by NoSQL databases? Is MongoDB a NoSQL database? explain.

At the present time, the internet is loaded with big data, big users, big complexity etc. and also becoming more complex day by day. NoSQL is answer of all these problems, It is not a traditional database management system, not even a relational database management system (RDBMS). NoSQL stands for "Not Only SQL". NoSQL is a type of database that can handle and sort all type of unstructured, messy and complicated data. It is just a new way to think about the database.

Yes. MongoDB is a NoSQL database.

**What are the different types of NoSQL databases? Give some example.**

NoSQL database can be classified as 4 basic types:

1. Key value store NoSQL database
2. Document store NoSQL database
3. Column store NoSQL database
4. Graph base NoSQL databse

There are many NoSQL databases. MongoDB, Cassandra, CouchBD, Hypertable, Redis, Riak, Neo4j, HBASE, Couchbase, MemcacheDB, Voldemort, RevenDB etc. are the examples of NoSQL databases.

### What is the difference between MongoDB and MySQL?

Although MongoDB and MySQL both are free and open source databases, there is a lot of difference between them in the term of data representation, relationship, transaction, querying data, schema design and definition, performance speed, normalization and many more. To compare MySQL with MongoDB is like a comparison between Relational and Non-relational databases.

### ****Define MongoDB.****

It is a document-oriented database that is used for high availability, easy scalability, and high performance. It supports the dynamic schema design.

**What makes MongoDB the best?**

MongoDB is considered to be the best NoSQL database because of its following features:

* Document-oriented (DO)
* High performance (HP)
* High availability (HA)
* Easy scalability
* Rich query language

### ****How to do transactions/locking in MongoDB?****

MongoDB does not use conventional locking with reduction as it is planned to be light, high-speed, and knowable in its presentation. It can be considered as parallel to the MySQL MyISAM auto entrust sculpt. With the simplest business sustain, performance is enhanced, particularly in a structure with numerous servers.

### ****If you remove an object attribute, is it deleted from the database?****

Yes, it is deleted. Hence, it is better to eliminate the attribute and then save the object again.

### ****How does MongoDB provide consistency?****

MongoDB uses the **reader–writer locks**, allowing simultaneous readers to access any supply like a database or a collection but always offering private access to single writes.

**Explain the structure of ObjectID in MongoDB.**

ObjectID is a 12-byte BSON type. These are:

* 4 bytes value representing seconds
* 3 byte machine identifier
* 2 byte process id
* 3 byte counter

### Is it true that MongoDB uses BSON to represent document structure?

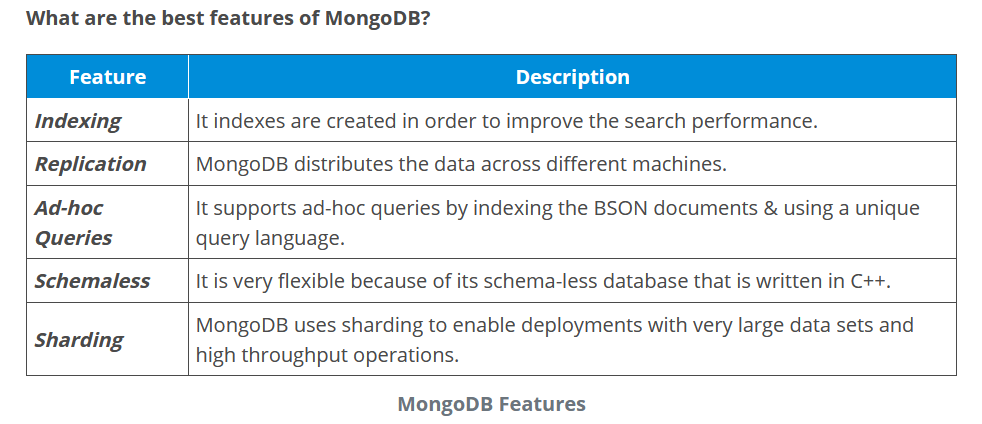
Yes.

### What are Indexes in MongoDB?

In MondoDB, Indexes are used to execute query efficiently. Without indexes, MongoDB must perform a collection scan, i.e. scan every document in a collection, to select those documents that match the query statement. If an appropriate index exists for a query, MongoDB can use the index to limit the number of documents it must inspect.

### By default, which index is created by MongoDB for every collection?

By default, the\_id collection is created for every collection by MongoDB.



**What is a ‘namespace’?**

MongoDB stores BSON objects in collections. The concatenation of the database name and the collection name (with a period in between) is called a ‘namespace’.

**Are null values allowed?**

Yes, but only for the members of an object. A null cannot be added to the database collection as it isn’t an object. But {}can be added.

**What’s a Master or Primary?**

This is a node/member which is currently the primary and processes all writes for the replica set. During a failover event in a replica set, a different member can become primary.

**What’s a Secondary or Slave?**

A secondary is a node/member which applies operations from the current primary. This is done by tailing the replication oplog (local.oplog.rs). Replication from primary to secondary is asynchronous, however, the secondary will try to stay as close to current as possible (often this is just a few milliseconds on a LAN).

**What are the disadvantages of MongoDB?**

* A 32-bit edition has 2GB data limit. After that it will corrupt the entire DB, including the existing data. A 64-bit edition won’t suffer from this bug/feature.
* Default installation of MongoDB has asynchronous and batch commits turned on. Meaning, it lies when asked to store something in DB and commits all changes in a batch at a later time in future. If there is a server crash or power failure, all those commits buffered in memory will be lost. This functionality can be disabled, but then it will perform as good as or worse than MySQL.
* MongoDB is only ideal for implementing things like analytics/caching where impact of small data loss is negligible.
* In MongoDB, it’s difficult to represent relationships between data so you end up doing that manually by creating another table to represent the relationship between rows in two or more tables.

**How to use the primary key in MongoDB?**

* **Ans.** \_id field is reserved for the primary key in MongoDB, and that is a unique value. If you don’t set anything to \_id it will automatically fill it with “MongoDB Id Object”. But you can put any unique info in that field.

**When should the data be normalized in MongoDB?**

**Ans.** It depends on your goals. Normalization will provide an update efficient data representation. Denormalization will make data reading efficiently.

In general, use embedded data models (denormalization) when:

* When you have “contains” relationships between entities.
* When you have one-to-many relationships between entities. In these relationships, the “many” or child documents always display with or are seen in the context of the “one” or parent documents.

In general, use normalized data models:

* When embedding leads to duplication of data but will not give sufficient read performance benefits to outweigh the implications of the duplication.
* To represent more complex many-to-many relationships.
* To model large hierarchical data sets.

**Comparison between MongoDB and Cassandra.**

**Ans.**

**MongoDB**:

* Data mode are the document
* Database scalability is read-only
* Query of the data is multi-index

**Cassandra**:

* Data mode are a big table like
* Database scalability is write only
* Query of the data by using scan or key

**Explain about replication and when should we use it?**

* **Ans.** [**Replication is the process of synchronizing data across multiple servers**](https://data-flair.training/blogs/mongodb-replication-and-sharding/) so that data is not lost in any condition. It gives redundancy and rises data availability with many copies of data on other database servers. We can use it for various purposes like for keeping the data safe, for high availability of data, disaster recovery and in no downtime for maintenance.

**How can MongoDB simulate join or subquery?**

* **Ans.** We are trying to figure out the best way to structure data in Mongo to simulate what would be a simple join or subquery in SQL.
* Say we have the classic Users and Posts example, with Users in one collection and Posts in another. We want to find all posts by users who’s city is “Mumbai”.
* We have simplified things in this question, in real-world scenario storing posts as an array in the user document won’t work as we have 1,000’s of “posts” per user constantly inserting.

Compare SQL databases and MongoDB at a high level.

SQL databases store data in form of tables, rows, columns and records. This data is stored in a pre-defined data model which is not very much flexible for today's real-world highly growing applications. MongoDB in contrast uses a flexible structure which can be easily modified and extended.

Does MongoDB support foreign key constraints?

No. MongoDB does not support such relationships.

Does MongoDB support ACID transaction management and locking functionalities?

No. MongoDB does not support default multi-document ACID transactions. However, MongoDB provides atomic operation on a single document.

How can you achieve primary key - foreign key relationships in MongoDB?

By default MongoDB does not support such primary key - foreign key relationships. However, we can achieve this concept by embedding one document inside another. Foe e.g. an address document can be embedded inside customer document.

What is Sharding in MongoDB? Explain.

Sharding is a method for storing data across multiple machines. MongoDB uses sharding to support deployments with very large data sets and high throughput operations.

What is a Storage Engine in MongoDB

A storage engine is the part of a database that is responsible for managing how data is stored on disk. For example, one storage engine might offer better performance for read-heavy workloads, and another might support a higher-throughput for write operations.

How does MongoDB provide concurrency?

MongoDB uses reader-writer locks that allow concurrent readers shared access to a resource, such as a database or collection, but give exclusive access to a single write operation.

### What is an embedded document?

**Ans:** Embedded documents specify the relationship between data that is written inside the body. The documents are received while the related data body is small.

### How does MongoDB offer consistency?

**Ans:** To provide consistency, MongoDB makes use of reader-writer locks to allow readers to simultaneously access any collection like a database but it always offers private access to single writers.