**What are NoSQL databases? What are the different types of NoSQL databases?**

A NoSQL database provides a mechanism for storage and retrieval of data that is modeled in means other than the tabular relations used in relational databases (like SQL, Oracle, etc.).

Types of NoSQL databases:

* Document Oriented
* Key Value
* Graph
* Column Oriented

**What kind of NoSQL database MongoDB is?**

MongoDB is a document oriented database. It stores data in the form of BSON structure based documents. These documents are stored in a collection.

**Which are the most important features of MongoDB?**

* Flexible data model in form of documents
* Agile and highly scalable database
* Faster than traditional databases
* Expressive query language

**What is a Namespace in MongoDB?**

A Namespace is the concatenation of the database name and collection name. For e.g. school.students with school as the database and students as the collection

**Which all languages can be used with MongoDB?**

Currently, MonggoDB provides official driver support for C, C++, C#, Java, Node.js, Perl, PHP, Python, Ruby, Scala, Go and Erlang. MongoDB can easily be used with any of these languages. There are some other community supported drivers too but the above mentioned ones are officially provided by MongoDB.

**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.

**How is MongoDB better than other SQL databases?**

MongoDB allows a highly flexible and scalable document structure. For e.g. one data document in MongoDB can have five columns and the other one in the same collection can have ten columns. Also, MongoDB database are faster as compared to SQL databases due to efficient indexing and storage techniques.

**Compare MongoDB and CouchDB at high level.**

Although both of these databases are document oriented, MongoDB is a better choice for applications which need dynamic queries and good performance on a very big database. On the other side, CouchDB is better used for applications with occasionally changing queries and pre-defined queries.

**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.

**Does MongoDB need a lot of RAM?**

No. MongoDB can be run even on a small amount of RAM. MongoDB dynamically allocates and de-allocates RAM based on the requirements of other processes.

**Does MongoDB pushes the writes to disk immediately or lazily?**

MongoDB pushes the data to disk lazily. It updates the immediately written to the journal but writing the data from journal to disk happens lazily.

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

ObjectID is a 12-byte BSON type with:

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

**MongoDB uses BSON to represent document structures. True or False?**

True

**If you remove a document from database, does MongoDB remove it from disk?**

Yes. Removing a document from database removes it from disk too.

Mention the command to insert a document in a database called school and collection called persons.

use school;

db.persons.insert({ name:"kadhir",dept:"CSE"})

**What are Indexes in MongoDB?**

Indexes support the efficient execution of queries in MongoDB. 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.

**How many indexes does MongoDB create by default for a new collection?**

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

**Can you create an index on an array field in MongoDB? If yes, what happens in this case?**

Yes. An array field can be indexed in MongoDB. In this case, MongoDB would index each value of the array.

**What is a covered query in MongoDB?**

A covered query is the one in which:

* fields used in the query are part of an index used in the query, and
* the fields returned in the results are in the same index

**Why is a covered query important?**

Since all the fields are covered in the index itself, MongoDB can match the query condition as well as return the result fields using the same index without looking inside the documents. Since indexes are stored in RAM or sequentially located on disk, such access is a lot faster.

**Does MongoDB provide a facility to do text searches? How?**

Yes. MongoDB supports creating text indexes to support text search inside string content. This was a new feature which can introduced in version 2.6.

**What happens if an index does not fit into RAM?**

If the indexes do not fit into RAM, MongoDB reads data from disk which is relatively very much slower than reading from RAM.

**Mention the command to list all the indexes on a particular collection.**

db.collection.getIndexes()

**At what interval does MongoDB write updates to the disk?**

By default configuration, MongoDB writes updates to the disk every 60 seconds. However, this is configurable with the commitIntervalMs and syncPeriodSecs options.

**How can you achieve transaction and locking in MongoDB?**

To achieve concepts of transaction and locking in MongoDB, we can use the nesting of documents, also called embedded documents. MongoDB supports atomic operations within a single document.

**What is Aggregation in MongoDB?**

Aggregations operations process data records and return computed results. Aggregation operations group values from multiple documents together, and can perform a variety of operations on the grouped data to return a single result. MongoDB provides three ways to perform aggregation: the aggregation pipeline, the map-reduce function, and single purpose aggregation methods and commands.

**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 Replication in MongoDB? Explain.**

Replication is the process of synchronizing data across multiple servers. Replication provides redundancy and increases data availability. With multiple copies of data on different database servers, replication protects a database from the loss of a single server. Replication also allows you to recover from hardware failure and service interruptions.

**What are Primary and Secondary Replica sets?**

Primary and master nodes are the nodes that can accept writes. MongoDB's replication is 'single-master:' only one node can accept write operations at a time.

**Secondary and slave nodes are read-only nodes that replicate from the primary.**

By default, MongoDB writes and reads data from both primary and secondary replica sets. True or False.

False. MongoDB writes data only to the primary replica set.

**Why are MongoDB data files large in size?**

MongoDB preallocates data files to reserve space and avoid file system fragmentation when you setup the server.

**When should we embed one document within another in MongoDB?**

You should consider embedding documents for:

* 'contains' relationships between entities
* One-to-many relationships
* Performance reasons

**Why MongoDB is not preferred over a 32-bit system?**

When running a 32-bit build of MongoDB, the total storage size for the server, including data and indexes, is 2 gigabytes. For this reason, do not deploy MongoDB to production on 32-bit machines.

**If you're running a 64-bit build of MongoDB, there's virtually no limit to storage size.**

**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.

**Which are the two storage engines used by MongoDB?**

MongoDB uses MMAPv1 and WiredTiger.

**What is the role of a profiler in MongoDB? Where does the writes all the data?**

The database profiler collects fine grained data about MongoDB write operations, cursors, database commands on a running mongod instance. You can enable profiling on a per-database or per-instance basis.

The database profiler writes all the data it collects to the system.profile collection, which is a capped collection.

**How does Journaling work in MongoDB?**

When running with journaling, MongoDB stores and applies write operations in memory and in the on-disk journal before the changes are present in the data files on disk. Writes to the journal are atomic, ensuring the consistency of the on-disk journal files. With journaling enabled, MongoDB creates a journal subdirectory within the directory defined by dbPath, which is /data/db by default.

Mention the command to check whether you are on the master server or not.Can you configure the cache size for MMAPv1? How?Can you configure the cache size for WiredTiger? How?How does MongoDB provide concurrency?How can you isolate your cursors from intervening with the write operations?

You can use the snapshot() method on a cursor to isolate the operation for a very specific case. snapshot() traverses the index on the \_id field and guarantees that the query will return each document no more than once.

**Can one MongoDB operation lock more than one databases? If yes, how?**

Yes. Operations like copyDatabase(), repairDatabase(), etc. can lock more than onne databases involved.

**How can concurrency affect replica sets primary?**

In replication, when MongoDB writes to a collection on the primary, MongoDB also writes to the primary's oplog, which is a special collection in the local database. Therefore, MongoDB must lock both the collection's database and the local database.

**What is GridFS?**

GridFS is a specification for storing and retrieving files that exceed the BSON-document size limit of 16MB. Instead of storing a file in a single document, GridFS divides a file into parts, or chunks, and stores each of those chunks as a separate document.

**Can you run multiple Javascript operations in a single mongod instance?**

Yes. The V8 JavaScript engine added in 2.4 allows multiple JavaScript operations to run at the same time.

**Which command can be used to provide various information on the query plans used by a MongoDB query?**

The explain() command can be used for this information. The possible modes are: 'queryPlanner', 'executionStats', and 'allPlansExecution'.

**What are the best features of MongoDB?**

|  |  |
| --- | --- |
| **MongoDB Features** | |
| **Feature** | **Description** |
| ***Indexing*** | It indexes are created in order to improve the search performance. |
| ***Replication*** | MongoDB distributes the data across different machines. |
| ***Ad-hoc Queries*** | It supports ad-hoc queries by indexing the BSON documents & using a unique query language. |
| ***Schemaless*** | It is very flexible because of its schema-less database that is written in C++. |
| ***Sharding*** | MongoDB uses sharding to enable deployments with very large data sets and high throughput operations. |

**When using replication, can some members use journaling and others not?**

Yes!

**Can journaling feature be used to perform safe hot backups?**

Yes!

**What is 32-bit nuances?**

There is an extra memory mapped file activity with journaling. This will further constrain the limited db size of 32-bit builds. For now, journaling by default is disabled on 32-bit systems.

**Will there be journal replay programs in case of incomplete entries (if there is a failure in the middle of one)?**

Each journal (group) write is consistent and won’t be replayed during recovery unless it is complete.

**What is the role of profiler in MongoDB?**

MongoDB includes a database profiler which shows performance characteristics of each operation against the database. With this profiler you can find queries (and write operations) which are slower than they should be and use this information for determining when an index is needed.

**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’.

**When an object attribute is removed, is it deleted from the store?**

Yes, you can remove the attribute and then re-save() the object.

**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.

**Does an update fsync to disk immediately?**

No. Writes to disk are lazy by default. A write may only hit the disk a couple of seconds later. For example, if the database receives thousand increments to an object within one second, it will only be flushed to disk once.*(Note: fsync options are available both at the command line and via getLastError\_old.)*

**How do I do transactions/locking?**

MongoDB does not use traditional locking or complex transactions with rollback, as it is designed to be light weight, fast and predictable in its performance. It can be thought of how analogous is to the MySQL’s MyISAMautocommit model. By keeping transaction support extremely simple, performance is enhanced, especially in a system that may run across many servers.

**Why are data files so large?**

MongoDB does aggressive preallocation of reserved space to avoid file system fragmentation.

**How long does replica set failover take?**

It may take 10-30 seconds for the primary to be declared down by the other members and a new primary to be elected. During this window of time, the cluster is down for primary operations i.e writes and strong consistent reads. However, eventually consistent queries may be executed to secondaries at any time (in slaveOk mode), including during this window.

**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).

**Is it required to call ‘getLastError’ to make a write durable?**

No. If ‘getLastError’ (aka ‘Safe Mode’) is not called, the server does exactly behave the way as if it has been called. The ‘getLastError’ call simply allows one to get a confirmation that the write operation was successfully committed. Of course, often you will want that confirmation, but the safety of the write and its durability is independent.

**Should you start out with Sharded or with a Non-Sharded MongoDB environment?**

We suggest starting with Non-Sharded for simplicity and quick startup, unless your initial data set will not fit on single servers. Upgrading to Sharded from Non-sharded is easy and seamless, so there is not a lot of advantage in setting up Sharding before your data set is large.

**How does Sharding work with replication?**

Each Shard is a logical collection of partitioned data. The shard could consist of a single server or a cluster of replicas. Using a replica set for each Shard is highly recommended.

**When will data be on more than one Shard?**

MongoDB Sharding is range-based. So all the objects in a collection lie into a chunk. Only when there is more than 1 chunk there is an option for multiple Shards to get data. Right now, the default chunk size is 64mb, so you need at least 64mb for migration.

**What happens when a document is updated on a chunk that is being migrated?**

The update will go through immediately on the old Shard and then the change will be replicated to the new Shard before ownership transfers.

**What happens when a Shard is down or slow when querying?**

If a Shard is down, the query will return an error unless the ‘Partial’ query options is set. If a shard is responding slowly, Mongos will wait for it.

**Can the old files in the ‘moveChunk’ directory be removed?**

Yes, these files are made as backups during normal Shard balancing operations. Once the operations are done then they can be deleted. The clean-up process is currently manual so this needs to be taken care of to free up space.

**How do you see the connections used by Mongos?**

The following command needs to be used: db.\_adminCommand(“connPoolStats”);

**If a ‘moveChunk’ fails, is it necessary to cleanup the partially moved docs?**

No, chunk moves are consistent and deterministic. The move will retry and when completed, the data will be only on the new Shard.

**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.