1. MongoDB is a cross-platform, non-relational document oriented database that provides, high performance, high availability, and easy scalability. MongoDB s name comes from humongous data.
2. The initial development of MongoDB began in 2007 when the company was building a platform as a service similar to window azure. MongoDB was developed by a NewYork based organization named 10gen which is now known as MongoDB Inc. Later in 2009, it is introduced in the market as an open source database server that was maintained and supported by MongoDB Inc.
3. The first ready production of MongoDB has been considered from version 1.4 which was released in March 2010. MongoDB 4.0.9 - was the latest and stable version which was released on Apr 16, 2019. You can get more information at the official site mongodb.com;
4. Many leading companies and other organizations choose open source technologies first. No license required - easy to experiment. So how can we see open source databases become more popular and dominate in new approaches.
5. MongoDB is an open source database and is a leader in popularity among NoSQL databases.
6. So what types of databases exist? There are many types: (Relational, Graph, Document Oriented and others) basically they are divided into SQL and NOSQL. NOSQL databases use model such as key value (Memcache), document (MongoDb), wide column (Cassandra) etc. while relational databases use tables;
7. So let a look what is document oriented database, and we do this by comparing relation database and mongodb. Any relational database has a typical schema design that shows number of tables and the relationship between these tables. While in MongoDB, there is no concept of relationship. The following table shows the relationship of RDBMS(relational) terminology with MongoDB.
   1. So the database concept is the same, it is container for something to store. Mongo db database is a physical container for collections, while relationals – for tables. Each database gets its own set of files on the file system. A single MongoDB server typically has multiple databases.
   2. Collection is a group of MongoDB documents. It is the equivalent of an RDBMS table. A collection exists within a single database. Collections do not enforce a schema. Documents within a collection can have different fields. Typically, all documents in a collection are of similar or related purpose.
   3. A record in MongoDB is a document, which is a data structure composed of field and value pairs. MongoDB documents are similar to JSON objects. The values of fields may include other documents, arrays, and arrays of documents. Documents have dynamic schema. Dynamic schema means that documents in the same collection do not need to have the same set of fields or structure, and common fields in a collection's documents may hold different types of data.
   4. \_id assures the uniqueness of every document. You can provide \_id while inserting the document. If you don’t provide then MongoDB provides a unique id for every document.
8. That's a little concrete example of what a document might look in MongoDB, like I said, no real schema is enforced in MongoDB at all, you can have different fields in every document if you want, but obviously it is not a good idea. Documents (i.e. objects) correspond to native data types in many programming languages. Embedded documents and arrays reduce need for expensive joins. Dynamic schema of the document supports fluent polymorphism.

This is an advantage that, all necessary information is stored in one document and you may do just one request to get it (this is called locality).But it is also a disadvantage. When we need to choose a small set of fields, for example, or change a single field, we have to read the entire document, that may lead to the spending of a large number of operations

1. Advantages of MongoDB over RDBMS

* Schema less − MongoDB is a document database in which one collection holds different documents. Number of fields, content and size of the document can differ from one document to another. Structure of a single object is clear. If you use Javascript for example and everything is simple for you because structure of the document likes JSON.

No complex joins. Deep query-ability. MongoDB supports dynamic queries on documents using a document-based query language that's nearly as powerful as SQL. You don’t need to know sql and all this stuff. Also MongoDb has such features as Data Aggregation, Text Search and Geospatial Queries. But keep in mind you still can't do efficient joins and can't deal with normalized data very efficiently. And one more point is transactions and consistency (ACID).the MongoDB documentation says : "Mongo support ACID transactions".

In real it doesn't support ACID, but support other minimum non-transactional guarantees.

* Ease of scale-out − MongoDB is easy to scale. by default, non-relational databases are split (or "shared") across many systems instead of only one. This makes it easier to improve performance at a lower cost.
* Conversion/mapping of application objects to database objects not needed.
* Uses internal memory for storing the (windowed) working set, enabling faster access of data

1. MongoDB has a single master architecture. The idea being that we want to have consistency over availability. But you can have these secondary databases that maintain copies over time from your primary database so as writes happen to your primary database. Those writes get replicated through an operation log to any secondary nodes that you might have attached to it.

MongoDB will automatically replicate those operations to those secondaries, so when the primary goes down one of these secondaries can take its place at the right way that replication chain works.

1. MongoDB provides horizontal scalability as part of its *core* functionality. [Sharding](https://docs.mongodb.com/manual/sharding/#sharding-introduction) distributes data across a cluster of machines. The way sharding works is that we actually have multiple replica sets where each replica set is responsible for some range of values on some indexed value in the database. So when you get starting to work it requires that you set up an index on some unique value on your collection and that index is used to actually balance the load of information among multiple replicas sets and then on each application server.

MongoDB supports many indices although you're still discouraged from doing more than two or three in a given collection and you can only have one that's used for sharding but you can actually set up things like full-text indices for doing efficient text searches across MongoDB.

1. Why Use MongoDB?

* Document Oriented Storage - Data is stored in the form of JSON style documents.
* Index on any attribute
* Replication and high availability
* Auto-sharding
* Rich queries
* Fast in-place updates
* Professional support by MongoDB

It actually has its own file system built-in as well called Grid FS that's kind of like HDFS where it's storing documents within go perform some MapReduce tasks on MongoDB.

1. Where to Use MongoDB?

* Big Data
* Content Management and Delivery
* Mobile and Social Infrastructure
* User Data Management
* Data Hub

MongoDB is really a good choice for things like storing big documents of information or text.

1. MongoDB is often used as a backend of large online games. Electronic Arts uses MongoDB for so many games. Why? Because scalability is important. If a game shoots well, it has to be scaled significantly more than expected.

At the end, I would like to say that there are no perfect database for all situations, it always depends on the task.