Learning Outcome

After completing this module, a student will be able to:

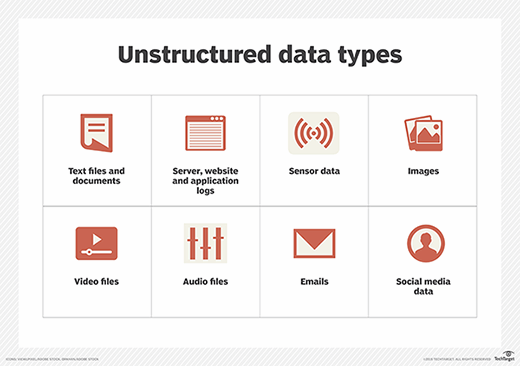
* Unstructured databases
* Structured vs unstructured data
* MongoDB fundamentals
* Databases
* Documents
* Collections
* MongoDB Atlas cluster
* MongoDB Atlas cluster connection and access remotely

# Unstructured databases

## What is Unstructured Data?

Unstructured data, typically categorized as qualitative data, cannot be processed and analyzed via conventional data tools and methods. Since unstructured data does not have a predefined data model, it is best managed in non-relational (NoSQL) databases. Another way to manage unstructured data is to use data lakes to preserve it in raw form.

The importance of unstructured data is rapidly increasing. Recent projections indicate that unstructured data is over 80% of all enterprise data, while 95% of businesses prioritize unstructured data management.



**Image:** Unstructured Database

**Reference:** https://cdn.ttgtmedia.com/rms/onlineImages/business\_analytics-unstructured\_data\_mobile.png

## Characteristics of Unstructured Data:

* Data neither conforms to a data model nor has any structure.
* Data cannot be stored in the form of rows and columns as in Databases
* Data does not follow any semantic or rules
* Data lacks any particular format or sequence
* Data has no easily identifiable structure
* Due to lack of identifiable structure, it cannot used by computer programs easily

## Pros and cons of unstructured data

Examples of unstructured data include text, mobile activity, social media posts, Internet of Things (IoT) sensor data, etc. Their benefits involve advantages in format, speed and storage, while liabilities revolve around expertise and available resources:

**Pros**

* **Native format:** Unstructured data, stored in its native format, remains undefined until needed. Its adaptability increases file formats in the database, which widens the data pool and enables data scientists to prepare and analyze only the data they need.
* **Fast accumulation rates:** Since there is no need to predefine the data, it can be collected quickly and easily.
* **Data lake storage:** Allows for massive storage and pay-as-you-use pricing, which cuts costs and eases scalability.

**Cons**

* **Requires expertise:** Due to its undefined/non-formatted nature, data science expertise is required to prepare and analyze unstructured data. This is beneficial to data analysts but alienates unspecialized business users who may not fully understand specialized data topics or how to utilize their data.
* **Specialized tools:** Specialized tools are required to manipulate unstructured data, which limits product choices for data managers.

## Unstructured data tools

* **MongoDB:** Uses flexible documents to process data for cross-platform applications and services.
* **DynamoDB:** Delivers single-digit millisecond performance at any scale via built-in security, in-memory caching and backup and restore.
* **Hadoop:** Provides distributed processing of large data sets using simple programming models and no formatting requirements.
* **Azure:** Enables agile cloud computing for creating and managing apps through Microsoft’s data centers.

## Use cases for unstructured data

* **Data mining:** Enables businesses to use unstructured data to identify consumer behavior, product sentiment, and purchasing patterns to better accommodate their customer base.
* **Predictive data analytics:** Alert businesses of important activity ahead of time so they can properly plan and accordingly adjust to significant market shifts.
* **Chatbots:** Perform text analysis to route customer questions to the appropriate answer sources.

## Examples of unstructured data

Unstructured data can be created by people or generated by machines.



**Image**: Example of Unstructured Data

**Reference:** https://wiki.atlan.com/content/images/2019/10/Unstructured-data-examples.png

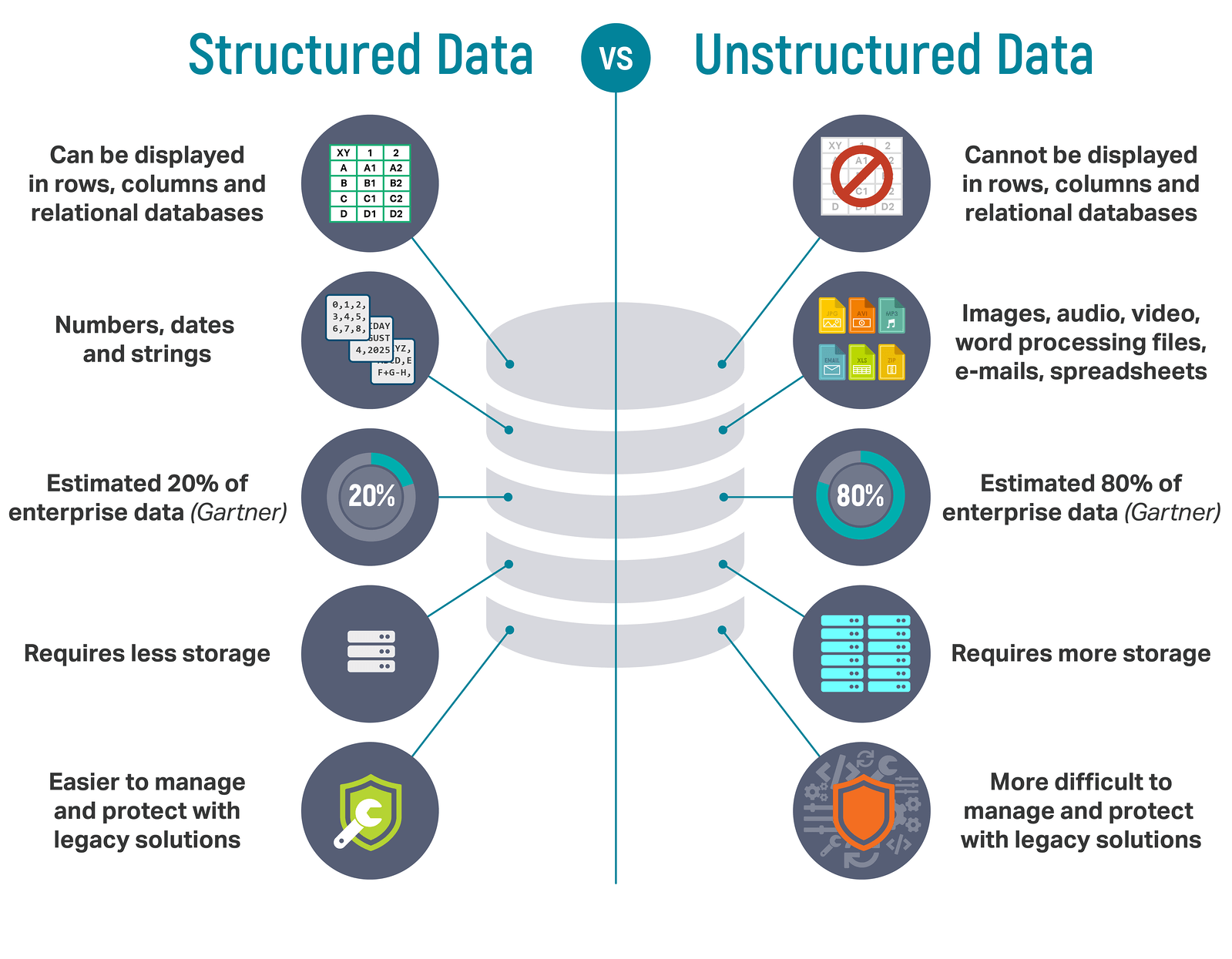
Here are some examples of the human-generated variety:

* **Email:** Email message fields are unstructured and cannot be parsed by traditional analytics tools. That said, email metadata affords it some structure, and explains why email is sometimes considered semi-structured data.
* **Text files:** This category includes word processing documents, spreadsheets, presentations, email, and log files.
* **Social media and websites:** Data from social networks like Twitter, LinkedIn, and Facebook, and websites such as Instagram, photo-sharing sites, and YouTube.
* **Mobile and communications data:** Text messages, phone recordings, collaboration software, Chat, and Instant Messaging.
* **Media:** Digital photos, audio, and video files

Here are some examples of unstructured data generated by machines:

* **Scientific data:** Oil and gas surveys, space exploration, seismic imagery, and atmospheric data.
* **Digital surveillance:** Reconnaissance photos and videos.
* **Satellite imagery:** Weather data, land forms, and military movements.

# Structured vs unstructured data



**Image:** Structured VS Unstructured Data

**Reference:** https://lawtomated.com/wp-content/uploads/2019/04/structuredVsUnstructuredIgneos.png

## Key characteristics

With structured data, every record adheres to a predefined data model; if incoming data fails to meet those definitions, it cannot be saved without correction or truncation. As a result, structured data may often be very text-heavy. This does have the advantage of being extremely easy to parse and search using conventional software.

Unstructured data is more ambiguous. Without a predefined data model, you can store a far broader range of rich data including images, sound, video, and text. As the scope for storage increases, and as data becomes more complex and dynamic, so too does the difficulty with which you can search and analyze that information. Thankfully, there are modern data management platforms, such as MongoDB Atlas, that make it easier to store and process large amounts of unstructured data.

## Data storage options

Because of its relative simplicity, structured data is well suited to the relative limitations of relational database systems. Large data estates can be housed in a data warehouse — as long as the information continues to meet the rigid database schema.

Unstructured data can be, and is, stored in a number of places. Specific applications like email servers often create their own data silo of unstructured information. Data warehouses and data lakes have become important for big data analytics, providing a way to increase overall capacity using low-cost commodity storage. For analyzing complex data types, or for advanced data analysis, NoSQL databases offer a way to more efficiently manage and search across disparate data sets.

## Where does the data come from?

Structured data is best suited to process-driven applications that rely on specific information presented in a known, consistent format. An inventory control system that maintains stock levels against product SKUs is an ideal example because it operates using concrete information. The logic built on top of the database may be complex, but the records themselves are very simple.

Unstructured data and applications powered by unstructured data tend to be more ambiguous; email clients that store messages of varying lengths that may include attachments. Or presentation software that blends text, graphics, and multimedia content. Potentially high value information is held in these assets, but it cannot be retrieved using regular text queries from a traditional relational database.

## Extracting value

The linear, controlled nature of structured data is best suited to statistical-type big data analytics using similarly structured query language (SQL). If you want to know which product line sells best during the summer months or which manufacturing component is likely to fail next, a regular relational database will perform adequately.

Unstructured data can also generate these insights — and a lot more. Going beyond raw statistics, unstructured data can (with the right NoSQL database), can provide more advanced insights, like customer sentiment. It can also provide enough structure so that non-text assets can be queried, allowing you to run facial recognition analysis from photographs for instance.

In the big data analytics environment, this additional layer of information provides much-needed context and insights that are not available from raw statistics and SQL-based, sanitized data sets. Further, the no-loss storage of unstructured data means that details remain intact even as your data needs and strategies change.

# MongoDB fundamentals

## Introduction to MongoDB

**Brief History of MongoDB**

* MongoDB was developed by Eliot Horowitz and Dwight Merriman in the year 2007
* When they experienced some scalability issues with the relational database while developing enterprise web applications at their company DoubleClick.
* According to Dwight Merriman, one of the developers of MongoDB, this name of the database was derived from the word humongous to support the idea of processing large amount of data.
* In 2009, MongoDB was made as an open source project, while the company offered commercial support services.
* Many companies started using MongoDB for its amazing features.
* The New York Times newspaper used MongoDB to build a web based application to submit the photos.
* In 2013, the company was officially named MongoDB Inc.

**What is MongoDB**

MongoDB is a cross-platform, document-oriented database that provides, high performance, high availability, and easy scalability. MongoDB works on concept of collection and document.

MongoDB is a popular NoSQL database that can store both structured and unstructured data. Founded in 2007 by Kevin P. Ryan, Dwight Merriman, and Eliot Horowitz in New York, the organization was initially called 10gen and was later renamed MongoDB—a word inspired by the term humongous.

**Database**

Database is a physical container for collections. Each database gets its own set of files on the file system. A single MongoDB server typically has multiple databases.

**Collection**

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.

**Document**

A document is a set of key-value pairs. 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.

The following table shows the relationship of RDBMS terminology with MongoDB.

|  |  |
| --- | --- |
| RDBMS | MongoDB |
| Database | Database |
| Table | Collection |
| Tuple/Row | Document |
| column | Field |
| Table Join | Embedded Documents |
| Primary Key | Primary Key (Default key \_id provided by MongoDB itself) |
| Database Server and Client | |
| mysqld/Oracle | mongod |
| mysql/sqlplus | mongo |

## Sample Document

Following example shows the document structure of a blog site, which is simply a comma separated key value pair.

{

\_id: ObjectId(7df78ad8902c)

title: 'MongoDB Overview',

description: 'MongoDB is no sql database',

by: 'tutorials point',

url: 'http://www.tutorialspoint.com',

tags: ['mongodb', 'database', 'NoSQL'],

likes: 100,

comments: [

{

user:'user1',

message: 'My first comment',

dateCreated: new Date(2011,1,20,2,15),

like: 0

},

{

user:'user2',

message: 'My second comments',

dateCreated: new Date(2011,1,25,7,45),

like: 5

}

]

}

\_id is a 12 bytes hexadecimal number which 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. These 12 bytes first 4 bytes for the current timestamp, next 3 bytes for machine id, next 2 bytes for process id of MongoDB server and remaining 3 bytes are simple incremental VALUE.

## Features

It provides both essential and extravagant features that are needed to store real-world big data. Its document-based design makes it easy to understand and use. It is built to be utilized for both experimental and real-world applications and is easier to set up and simpler to manage than most of the other NoSQL databases. Its intuitive syntax for queries and commands makes it easy to learn.

**The following list explores these features in detail:**

* **Flexible and Dynamic Schema:** MongoDB allows a flexible schema for your database. A flexible schema allows variance in fields in different documents. In simple terms, each record in the database may or may not have the same number of attributes. It addresses the need for storing evolving data without making any changes to the schema itself.
* **Rich Query Language:** MongoDB supports intuitive and rich query language, which means simple yet powerful queries. It comes with a rich aggregation framework that allows you to group and filter data as required. It also has built-in support for general-purpose text search and specific purposes like geospatial searches.
* **Multi-Document ACID Transactions:** Atomicity, Consistency, Integrity, and Durability (ACID) are features that allow your data to be stored and updated to maintain its accuracy. Transactions are used to combine operations that are required to be executed together. MongoDB supports ACID in a single document and multi-document transactions.
* Atomicity means all or nothing, which means either all operations are a part of a transaction as it happens or none of them are. This means that if one of the operations fails, then all the executed operations are rolled back to leave the data affected by transaction operation in the state it was in before the transaction started.
* Consistency in a transaction means keeping the data consistent as per the rules defined for the database. If a transaction breaks any database consistency rules, then it must be rolled back.
* Isolation enforces running transactions in isolation, which means that the transactions do not partially commit the data and any values outside the transactions change only after all the operations are executed and are fully committed.
* Durability ensures that the changes are committed by the transaction. So, if a transaction has executed then the database will ensure the changes are committed even if there is a system crash.
* **High Performance:** MongoDB provides high performance using embedded data models to reduce disk I/O usage. Also, extensive support for indexing on different kinds of data makes queries faster. Indexing is a mechanism to maintain relevant data pointers in an index just like an index in a book.
* **High Availability:** MongoDB supports distributed clusters with a minimum of three nodes. A cluster refers to a database deployment that uses multiple nodes/machines for data storage and retrieval. Failovers are automatic, and data is replicated on secondary nodes asynchronously.
* **Scalability:** MongoDB provides a way to scale your databases horizontally across hundreds of nodes. So, for all your big data needs, MongoDB is the perfect solution. With this, we have looked at some of the essential features of MongoDB.Note.

## 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.
* 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.
* Tuning.
* **Ease of scale-out** − MongoDB is easy to scale.
* Conversion/mapping of application objects to database objects not needed.
* Uses internal memory for storing the (windowed) working set, enabling faster access of data.

## 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

## Where to Use MongoDB?

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

## MongoDB Installation

To install MongoDB in you physical machine use this link: <https://docs.mongodb.com/manual/installation/>

## Data Types

MongoDB stores documents on disk in the BSON serialization format. BSON is a binary representation of JSON documents, though BSON data format provides more data types than JSON. The mongo JavaScript shell and the MongoDB language drivers translate between BSON and the language-specific document representation.

BSON supports the following data types as values in documents. Each data type has a corresponding number (an integer ID number from 1 to 255) that can be used with the $type operator to query documents by BSON type.

**MongoDB Data Types and Corresponding ID Number**

|  |  |  |
| --- | --- | --- |
| Type | Description | Number |
| Double | Represents a float value. | 1 |
| String | BSON4 strings are UTF-8. In general, drivers for each programming language convert from the language’s string format to UTF-8 when serializing and deserializing BSON. This makes it possible to store most international characters in BSON strings with ease. [1] In addition, MongoDB $regex queries support UTF-8 in the regex string. | 2 |
| Object | Represents an embedded document. | 3 |
| Array | Sets or lists of values can be represented as arrays: | 4 |
| Binary data | Binary data is a string of arbitrary bytes, it cannot be manipulated from the shell. | 5 |
| Object id | ObjectIds (MongoDB document identifier, equivalent to a Primary key) are: small, likely unique, fast to generate, and ordered. These values consists of 12-bytes, where the first four bytes are a timestamp that reflect the ObjectId’s creation. | 7 |
| Boolean | A logical true or false. Use to evaluate whether a condition is true or false | 8 |
| Date | BSON Date is a 64-bit integer that represents the number of milliseconds since the Unix epoch (Jan 1, 1970). This results in a representable date range of about 290 million years into the past and future. | 9 |
| Null | It represents both a null value and a nonexistent field. | 10 |
| Regular Expression | RegExp maps directly to a Javascript RegExp | 11 |
| JavaScript |  | 13 |
| Symbol | Not supported by the shell. If the shell gets a symbol from the database, it will convert it into a string. | 14 |
| JavaScript (with scope) |  | 15 |
| 32-bit integer | Numbers without decimal points will be saved as 32-bit integers. | 16 |
| Timestamp | BSON has a special timestamp type for internal MongoDB use and is not associated with the regular Date type. Timestamp values are a 64 bit value where :   * the first 32 bits are a time\_t value (seconds since the Unix epoch). * the second 32 bits are an incrementing ordinal for operations within a given second. |  |
| 64-bit integer | Numbers without a decimal point will be saved and returned as 64-bit integers. | 18 |
| Min key | MinKey compare less than all other possible BSON element values, respectively, and exist primarily for internal use. | 255 |
| Max key | MaxKey compare greater than all other possible BSON element values, respectively, and exist primarily for internal use. | 127 |

**Comparing values of different BSON types**

When comparing values of different BSON types, MongoDB uses the following comparison order, from lowest to highest:

|  |  |
| --- | --- |
| Order | Data Types |
| 1 | MinKey (internal type) |
| 2 | Null |
| 3 | Numbers (ints, longs, doubles) |
| 4 | Symbol, String |
| 5 | Object |
| 6 | Array |
| 7 | BinData |
| 8 | ObjectId |
| 9 | Boolean |
| 10 | Date, Timestamp |
| 11 | Regular Expression |
| 12 | MaxKey (internal type) |

# Databases

For storing data in a MongoDB, you need to create a database first. It will allow you to systematically organize your data so that it can be retrieved as per requirement. If you wish to delete a database, MongoDB also allows you to delete that. In this chapter, you will learn how to create and delete a database in MongoDB.

A number of databases can be run on a single MongoDB server. Default database of MongoDB is 'db', which is stored within data folder.

MongoDB can create databases on the fly. It is not required to create a database before you start working with it.

**View the list of Databases in MongoDB**

"show dbs" command provides you with a list of all the databases.



**Image:** show dbs command

**Reference:** https://www.w3resource.com/w3r\_images/show-dbs-command.png

Run 'db' command to refer to the current database object or connection.

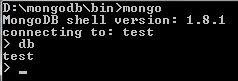


Image: db command

Reference: https://www.w3resource.com/w3r\_images/db-command.png

**“use” command for creating database in MsongoDB**

You can make use of the "use" command followed by the database\_name for creating a database. This command will tell the MongoDB client to create a database by this name if there is no database exists by this name. Otherwise, this command will return the existing database that has

the name.

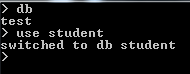


Image: use command

Reference: <https://www.w3resource.com/w3r_images/use-command.png>

In the above command, 'student' is the database we want to select.

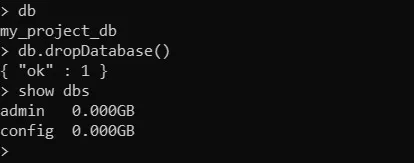
Database names can be almost any character in the ASCII range. But they can't contain an empty string, a dot (i.e. ".") or " ".

Since it is reserved, "system" can't be used as a database name.

A database name can contain "$".

**Deleting a Database in MongoDB**

If you are familiar with SQL, then you must have heard about the drop command. The concept of drop in SQL is used to delete the entire database or just the table, i.e., it destroys the objects like an existing database. In MongoDB, the dropDatabase command is implemented for a similar purpose. This also helps in deleting the connected data files of that database. For operating this command, you have to reside on the current database.



**Image:** delete command

**Reference:** <https://www.w3schools.in/wp-content/uploads/2019/06/mongodb_drop_database_command-1.jpg?ezimgfmt=rs:414x163/rscb7/ng:webp/ngcb7>

# Documents

The document is the unit of storing data in a MongoDB database.

document use JSON (JavaScript Object Notation, is a lightweight, thoroughly explorable format used to interchange data between various applications) style for storing data.

A simple example of a JSON document is as follows

{ site : "w3resource.com"

Often, the term "object" is used to refer a document.

Documents are analogous to the records of an RDBMS. Insert, update, and delete operations can be performed on a collection. The following table will help you to understand the concept more easily :

|  |  |
| --- | --- |
| RDBMS | MongoDB |
| Table | Collection |
| Column | Key |
| Value | Value |
| Records / Rows | Document / Object |

The following table shows the various datatypes which may be used in MongoDB.

|  |  |
| --- | --- |
| Data Types | Description |
| string | May be an empty string or a combination of characters. |
| integer | Digits. |
| boolean | Logical values True or False. |
| double | A type of floating point number. |
| null | Not zero, not empty. |
| array | A list of values. |
| object | An entity which can be used in programming. May be a value, variable, function, or data structure. |
| timestamp | A 64 bit value referring to a time and unique on a single "mongod" instance. The first 32 bit of this value refers to seconds since the UTC January 1, 1970. And last 32 bits refer to the incrementing ordinal for operations within a given second. |
| Internationalized Strings | UTF-8 for strings. |
| Object IDs | Every MongoDB object or document must have an Object ID which is unique. This is a BSON(Binary JavaScript Object Notation, which is the binary interpretation of JSON) object id, a 12-byte binary value which has a very rare chance of getting duplicated. This id consists of a 4-byte timestamp (seconds since epoch), a 3-byte machine id, a 2-byte process id, and a 3-byte counter. |

# Collections

A collection may store a number of documents. A collection is analogous to a table of an RDBMS.

A collection may store documents those who are not same in structure. This is possible because MongoDB is a Schema-free database. In a relational database like MySQL, a schema defines the organization / structure of data in a database. MongoDB does not require such a set of formula defining structure of data. So, it is quite possible to store documents of varying structures in a collection. Practically, you don't need to define a column and it's datatype unlike in RDBMS, while working with MongoDB.

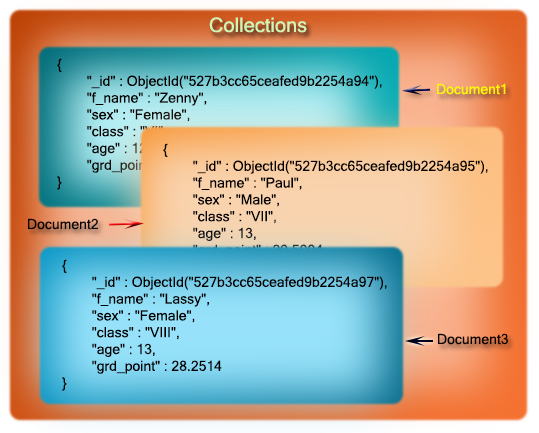
In the following code, it is shown that two MongoDB documents, belongs to same collection, storing data of different structures.

{"tutorial" : "NoSQL"}

{"topic\_id" : 7}

A collection is created, when the first document is inserted.

**Pictorial Presentation:** Collections and Documents



**Image:** Collections

**Reference:** https://www.w3resource.com/w3r\_images/mongodb-document-collection.png

## Valid collection names

* Collection names must begin with letters or an underscore.
* A Collection name may contain numbers
* You can't use "$" character within the name of a collection. "$" is reserved.
* A Collection name must not exceed 128 characters. It will be nice if you keep it within 80/90 characters.
* Using a "." (dot) notation, collections can be organized in named groups. For example, tutorials.php and tutorials.javascript both belong to tutorials. This mechanism is called as collection namespace which is for user primarily. Databases don't have much to do with it.

Following is how to use it programmatically:

db.tutorials.php.findOne()

## Capped collections

Imagine that you want to log the activities happening with an application. you want to store data in the same order it is inserted. MongoDB offers Capped collections for doing so.

Capped collections are collections which can store data in the same order it is inserted.

It is very fixed size, high-performance and "auto-FIFO age-Out". That is, when the allotted space is fully utilized, newly added objects (documents) will replace the older ones in the same order it is inserted.

Since data is stored in the natural order, that is the order it is inserted, while retrieving data, no ordering is required, unless you want to reverse the order.

New objects can be inserted into a capped collection.

Existing objects can be updated.

But you can't remove an individual object from the capped collection. Using drop command, you have to remove all the documents. After the drop, you have to recreate the capped collection.

Presently, the maximum size for a capped collection is 1e9(i.e. 1X109) for 32-bit machines. For 64 bit machines, there is no theoretical limit. Practically, it can be extended till your system resources permit.

Capped collections can be used for logging, caching and auto archiving.

## Use number of collections instead of one

This omits the requirement if creating index since you are not storing some repeating data on each object.

If applied to a suitable situation, it can enhance the performance.

## Metadata

Information about a database is stored in certain collections. They are grouped in system namespace, as

dbname.system.\*

The following table shows the collections and what they store

|  |  |
| --- | --- |
| Collections with namespace | Description |
| dbname.system.namespaces | list of all namespaces |
| dbname.system.indexes | list of all indexes |
| dbname.system.profile | stores database profiling information |
| dbname.system.users | list of users who may access the database |
| dbname.local.sources | stores replica slave configuration data and state |
| dbname.local.sources | stores replica slave configuration data and state |

There are two more options to store metadata:

database.ns files stores additional namespace / index metadata if exists.

# MongoDB Atlas cluster

## MongoDB Atlas

MongoDB Atlas is the DBaaS offering from MongoDB Inc. It allows you to provision a database on the cloud as a service, which can be used for your applications from anywhere. Atlas uses cloud infrastructures from different cloud vendors. You can choose the cloud vendor on which you want to deploy your database. Like any other managed service, you get the benefits of highly available secured environments with low or no maintenance needed.

## MongoDB Atlas Benefits

Let us look at some of the benefits of MongoDB Atlas.

* **Simple Setup:** The database setup on Atlas is easy and can be done in just a few steps. Atlas runs a variety of automated tasks behind the scenes to set up your multi-node cluster.
* **Guaranteed Availability:** Atlas deploys at least three data nodes or servers per replica set. Each node is deployed in a separate availability zone (Amazon Web Services (AWS)), fault domains (Microsoft Azure), or zones (Google Cloud Platform (GCP)). This allows a highly available setup and continuous uptime in case of outages or routine updates.
* **Global Presence:** MongoDB Atlas is available across different regions in the AWS, GCP, and Microsoft Azure clouds. The support for different regions allows you to pick a region closer to you for low latency read and write.
* **Optimal Performance:** The founders of MongoDB manage Atlas, and they utilize their expertise and experience to keep the databases in Atlas running optimally. Also, single-click upgrades are available for upgrading to the latest versions of MongoDB.
* **Highly Secured:** Security best practices are implemented by default, such as a separate VPC (virtual private cloud), network encryption, access controls, and firewalls to restrict access.
* **Automated Backups:** You can configure automated backups with customizable schedules and data retention policies. Secure backups and restores are available for switching between different versions of your database.

## Advantages of MongoDB Atlas

* **Global clusters for world-class applications:** Using MongoDB Atlas, we are free to choose the cloud partner and ecosystem that fit our business strategy.
* **Secure for sensitive data:** It offers built-in security controls for all our data. It enables enterprise-grade features to integrate with our existing security protocols and compliance standard.
* **Designed for developer productivity:** MongoDB Atlas moves faster with general tools to work with our data and a platform of services that makes it easy to build, secure, and extend applications that run on MongoDB.
* **Reliable for mission-critical workload:** It is built with distributed fault tolerance and automated data recovery.
* **Built for optimal performance:** It makes it easy to scale our databases in any direction. We can get more out of our existing resources with performance optimization tools and real-time visibility into database metrics.
* **Managed for operational efficiency:** It comes with built-in operational best practices, so we can focus on delivering business value and accelerating application development instead of managing databases.

## What are clusters in MongoDB?

In the context of MongoDB, “cluster” is the word usually used for either a replica set or a sharded cluster. A replica set is the replication of a group of MongoDB servers that hold copies of the same data; this is a fundamental property for production deployments as it ensures high availability and redundancy, which are crucial features to have in place in case of failovers and planned maintenance periods.

A sharded cluster is also commonly known as horizontal scaling, where data is distributed across many servers.

The main purpose of sharded MongoDB is to scale reads and writes along multiple shards.

## What is MongoDB Atlas Cluster?

MongoDB Atlas Cluster is a NoSQL Database-as-a-Service offering in the public cloud (available in Microsoft Azure, Google Cloud Platform, Amazon Web Services). This is a managed MongoDB service, and with just a few clicks, you can set up a working MongoDB cluster, accessible from your favorite web browser.

You don’t need to install any software on your workstation as you can connect to MongoDB directly from the web user interface as well as inspect, query, and visualize data.

Alternatively, if you prefer working with the command line, you can connect using the mongo shell. To do this, you’ll need to configure the firewall from the web portal to accept your IP. From the homepage, navigate to Security and then Network Access. Finally, click on “Add IP Address” and add your IP:

Then, substitute the following configuration settings (MongoDB Atlas cluster name, database name, and username) in the mongo shell command line window. For example:

mongo "mongodb+srv://<clustername>.nupbd.mongodb.net/<dbname>" --username <username>

Note: When using the mongo shell like above, you will be prompted to type the password that you submitted when the MongoDB deployment was created.

# MongoDB Atlas cluster connection and access remotely

## Create an Atlas Account and deploying a Free Tier Cluster

**Step 1:** Go to <https://www.mongodb.com/cloud/atlas>

to register for an Atlas account to host your data.

**Step 2:** When you click on Start Free, you will be redirected to the Registration form for an account on the MongoDB Atlas.

**Step 3:** Select Starter Clusters and click create a Cluster. The Starter cluster includes the M0, M2, and M5 cluster tiers. These low-cost clusters are suitable for users who are learning MongoDB or developing small proof -of- concept applications.

**Step 4:** Select your preferred Cloud Provider & Region. It supports M0 Free Tier clusters on Amazon Web Services (AWS), Google Cloud Platform (GCP), and Microsoft Azure. The regions that support M0 Free tier clusters are marked with the "Free Tier Available" label.

**Step 5:** Select M0 Sandbox for cluster tier: Selecting M0 automatically locks the remaining configuration options. If you can't select the M0 cluster tier, return to the previous step and choose a Cloud Provider & Region that supports M0 Free Tier clusters.

**Step 6:** Enter a name for your cluster in the Cluster Name field; you can enter any name for your cluster. The cluster name contains ASCII letters, numbers, and hyphens.

**Step 7:** Click on Create Cluster to deploy the cluster. Once you deploy your cluster, it can take up to 5-10 min for your cluster to provision and become ready to use.

**Step 8:** Once we register, Atlas automatically creates a default organization and project where we can deploy our first cluster. We can add additional organizations and projects later.

**How to Whitelist your Connection IP Address**

An IP address is a unique numeric identifier for a device connecting to a network. In Atlas, we can only connect to a cluster from a trusted IP address. Within Atlas, we can create an inventory of trusted IP addresses, mentioned as a Whitelist, which may be used to connect to our cluster and access our data.

We must add our IP address to the whitelist before we can connect to our cluster — the following steps we should take to whitelist our connection IP address.

**Step 1:** First, click on the Connect button from our cluster view.

**Step 2:** Configure your Whitelist entry. In the Whitelist your connection IP address step, click Add our Current IP Address.

**Step 3:** Click Add IP Address.

## Creating a MongoDB User for our Cluster

For security purposes, Atlas requires clients to authenticate as MongoDB users to access the cluster. We must create a MongoDB user to access our cluster. MongoDB users are separate from Atlas users:

MongoDB users can access a database hosted in Atlas.

Atlas users can log into Atlas but do not have access to MongoDB databases.

**Step 1:** Open the Connect dialogue from our Cluster view.

**Step 2:** In the Create a MongoDB User step of the dialog, enter a Username and a password for our MongoDB user.

**Step 3:** Finally, click on Create MongoDB User button.

## Connecting to Cluster

We can connect to our cluster in several ways. We will see how we can connect our cluster using the mongo shell driver.

Following are the steps to download and install the mongo shell driver.

**Step 1:** Open the Connect dialog from your cluster.

**Step 2:** Click connect with the Mongo Shell.

**Step 3:** Click on "I do not have the Mongo Shell installed" and select your operating system from the dropdown.

**Step 4:** Click on download the Mongo Shell.

**Step 5:** To run a Mongo Shell from your PC, you will need to add the Shell to your system path.

**Step 6:** To see if you have correctly added the Mongo Shell to your system-path, run the following command in your terminal.

mongo - - version

**Connect to your Atlas Cluster**

**Step 7:** Click on "I have the Mongo Shell Installed" and select your Mongo Shell version from the drop-down menu.

**Step 8:** Now, copy the provided connection string to your clipboard, then paste and run your connection string in your terminal.

**Step 9:** Now, you will be prompted to enter the password you specified when you created your MongoDB user in Atlas.

You are now will be connected to your Atlas cluster within the Mongo Shell.

# Reference

1. <https://www.ibm.com/cloud/blog/structured-vs-unstructured-data>
2. <https://www.geeksforgeeks.org/what-is-unstructured-data/>
3. <https://www.mongodb.com/unstructured-data#:~:text=Unstructured%20data%20is%20information%20that,common%20types%20of%20unstructured%20content>.
4. <https://www.mongodb.com/unstructured-data/structured-vs-unstructured>
5. <https://www.packtpub.com/product/mongodb-fundamentals/9781839210648>
6. <https://www.tutorialspoint.com/mongodb/mongodb_advantages.htm>
7. <https://www.w3resource.com/mongodb/databases-documents-collections.php>
8. <https://docs.mongodb.com/manual/installation/>
9. <https://www.packtpub.com/product/mongodb-fundamentals/9781839210648>
10. <https://www.javatpoint.com/mongodb-atlas>