Databases

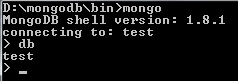
A number of databases can be run on a single MongoDB server. Default database of MongoDB is 'db', which is stored within data folder.

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

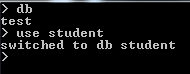
"show dbs" command provides you with a list of all the databases.



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



To connect to a particular database, run use command.



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

w3resource MongoDB tutorial has a separate page dedicated to commands related to creation and management of the database.

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 "$".

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.

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.

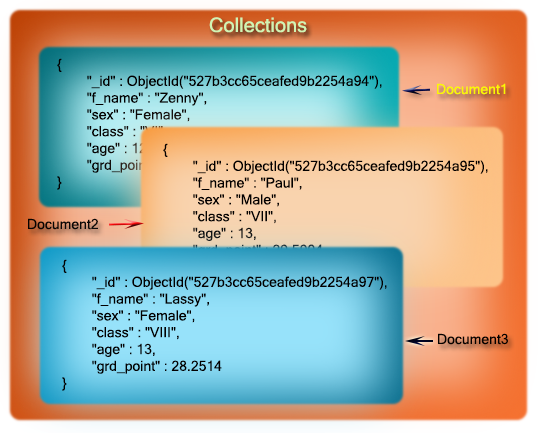
In the following code, it is shown that two MongoDB documents, belongs to same collection, storing data of different structures.

{"tutorial" : "NoSQL"}

{"topic\_id" : 7}

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

Pictorial Presentation : Collections and Documents



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()

Copy

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.

Metadata

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

dbname.system.\*

Copy

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.

Information on the structure of a stored object is stored within the object itself.

**MONGODB QUERY AND PROJECTION OPERATORS**

Comparison Query operators

|  |  |
| --- | --- |
| **Name** | **Description** |
| [$gt](https://www.w3resource.com/mongodb/mongodb-conditional-operators.php) | Matches values that are greater than the value specified in the query. |
| [$lt](https://www.w3resource.com/mongodb/mongodb-conditional-operators.php) | Matches values that are less than the value specified in the query. |
| [$gte](https://www.w3resource.com/mongodb/mongodb-conditional-operators.php) | Matches values that are greater than or equal to the value specified in the query. |
| [$lte](https://www.w3resource.com/mongodb/mongodb-conditional-operators.php) | Matches values that are less than or equal to the value specified in the query. |
| [$ne](https://www.w3resource.com/mongodb/mongodb-ne-operators.php) | Matches all values that are not equal to the value specified in the query. |
| [$in](https://www.w3resource.com/mongodb/mongodb-in-operators.php) | Matches any of the values that exist in an array specified in the query. |
| [$nin](https://www.w3resource.com/mongodb/mongodb-in-operators.php) | Matches values that do not exist in an array specified in the query. |

Logical Query operators

|  |  |
| --- | --- |
| **Name** | **Description** |
| [$and](https://www.w3resource.com/mongodb/mongodb-logical-and-not-operators.php) | Joins query clauses with a logical AND returns all documents that match the conditions of both clauses. |
| [$not](https://www.w3resource.com/mongodb/mongodb-logical-and-not-operators.php) | Inverts the effect of a query expression and returns documents that do not match the query expression. |
| [$or](https://www.w3resource.com/mongodb/mongodb-or-nor-operators.php) | Joins query clauses with a logical OR returns all documents that match the conditions of either clause. |
| [$nor](https://www.w3resource.com/mongodb/mongodb-or-nor-operators.php) | Joins query clauses with a logical NOR returns all documents that fail to match both clauses. |

$and example:

>db.student.find({$and:[{"sex":"Male"},{"grd\_point":{ $gte: 31 }},{"class":"VI"}]}).pretty();

SQL equivalent command is

SELECT \*

FROM student

WHERE sex="Male" AND grd\_point>=31 AND class="VI";

$not example

With regex

db.restaurants.find({ name :{ $not:/.\*mon.\*/ }}).count()

>db.student.find( {"sex": { $not: /^M.\*/}}).pretty();

SELECT \*

FROM student

WHERE sex NOT LIKE 'M%';

>db.student.find( {"age": { $not: {$lt : 12}}}).pretty();

SQL equivalent command is

SELECT \*

FROM student

WHERE sex="Male" AND grd\_point>=31 AND class="VI";

Evaluation Query operators

|  |  |
| --- | --- |
| **Name** | **Description** |
| [$mod](https://www.w3resource.com/mongodb/mongodb-mod-operators.php) | The $mod operator is used to help the user to retrieve documents from a collection for a specific field that consist an even or odd number.  The $mod operator works only on integer values and not a strings type numeric value. |
| [$regex](https://www.w3resource.com/mongodb/mongodb-regex-operators.php) | Selects documents where values match a specified regular expression. |
| [$where](https://www.w3resource.com/mongodb/mongodb-where-operators.php) | Matches documents that satisfy a JavaScript expression. |

Query operator Array

|  |  |
| --- | --- |
| **Name** | Description |
| [$all](https://www.w3resource.com/mongodb/mongodb-all-operators.php) | Matches arrays that contain all elements specified in the query. |
| [$elemMatch](https://www.w3resource.com/mongodb/mongodb-elemmatch-operators.php) | The MongoDB $elemMatch operator matches more than one component within an array element. |
| [$size](https://www.w3resource.com/mongodb/mongodb-size-operators.php) | Selects documents if the array field is a specified size. |

Projection operator

|  |  |
| --- | --- |
| **Name** | **Description** |
| [$](https://www.w3resource.com/mongodb/mongodb-$-projection-operators.php) | Projects the first element in an array that matches the query condition. |
| [$elemMatch(projection)](https://www.w3resource.com/mongodb/mongodb-elemmatch-projection-operators.php) | Projects the first element in an array that matches the specified $elemMatch condition. |
| [$slice](https://www.w3resource.com/mongodb/mongodb-slice-operators.php) | Limits the number of elements projected from an array. Supports skip and limit slices. |