Indexing

Index Creation

`db.coll.ensureIndex(key_pattern, options)`

Create an index on collection **coll** with the given key pattern and options.

Indexing Key Patterns with Sample Queries

`{username: 1}` `Ex: db.users.find({username: 'smith'});`	Simple index on **username**.
`{last_name: 1, last_login: -1}` `Ex: db.users.find({last_name: 'jones'}).sort({last_login: -1})`	Compound index with **last_name** ascending and **last_login** descending. Note that key order on compound indexes matters.
`{coord: '2d'}` `Ex: db.places.find({coord: {$near: [50, 50]}})`	Geospatial index, where **coord** is a coordinate **(x,y)** where -180 < **x**, **y** < 180. Note that **$near** queries return the closest points to the given coordinate.
`{loc: '2dsphere'}` `db.places.find({coord: {$near: {$geometry: {type: "Point", coordinates: [ 125, 90 ]}}}})`	Geospatial index where the **loc** field stores GeoJSON data. Note that you should always store coordinates in the following order: longitude, latitude. Valid longitude values are between -180 and 180. Valid latitude values are between -90 and 90.

Index Creation Options

`{unique: true}`	Create a unique index. To check insertion failures, you must use your driver's safe mode.
`{dropDups: true}`	Use with the **unique** option. Drop documents with duplicate values for the given key pattern on index creation.
`{background: true}`	Create this index in the background; useful when you need to minimize index creation performance impact.
`{sparse: true}`	Create entries in the index only for documents having the index key.
`{name: 'foo'}`	Specify a custom name for this index. If not specified, the name will be derived from the key pattern.

Examples

`db.users.ensureIndex({username: 1}, {unique: true})`	Create a unique index on **`username`**.
`db.products.ensureIndex({category: 1, price: -1}, {background: true})`	Create a compound index on **`category`** and **`price`** and build it in the background.
`db.places.ensureIndex({loc: '2dsphere'})`	Create a **`2dsphere`** geospatial index on **`loc`**.

Administration

`db.users.getIndexes()`	Get a list of all indexes on the **`users`** collection.
`db.users.totalIndexSize()`	Get the number of bytes allocated by indexes for the **`users`** collection.
`db.users.reIndex()`	Rebuild all indexes on this collection.
`db.users.dropIndex({x: 1, y: -1})`	Drop the index with key pattern **`{x: 1, y: -1}`**. Use **`db.users.dropIndexes()`** to drop all indexes on the **`users`** collection.

Tips

You can use a compound index on **`{username: 1, date: 1}`** for the following queries:

```
db.users.find({username: "Jones"});
db.users.find({username: /^Jones/});
db.users.find({username: "Jones", date: new Date()});
db.users.find({username: "Jones"}).sort({date: -1});
db.users.find({}).sort({username: 1, date: 1}).limit(100);
```

Note that with this index, a separate single-key index on **`{username: 1}`** is unnecessary.

Created and distributed by MongoDB, Inc. For more information or to download MongoDB, visit mongodb.org or mongodb.com.