mongoDB

www.mongodb.org
@mongodb

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}
db.users.find({username: 'smith'});
```

Simple index.

```
{name: 1, last_login: -1}
db.users.find({last_name: 'jones'}}).
sort({last_login: -1})
```

Compound index with *name* ascending and *last_login* descending. (Note that key order on compound indexes matters).

```
{coord: '2d'}
db.places.find( { coord : {
$near : [50, 50] } } )
```

Geospatial index, where *coord* is a coordinate (x,y) where -180 < x, y < 180 *$near* queries return the closest points to the given coordinate.

Index creation options

`{unique: true}` **Create a unique index. To check insertion failures, 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.**

`{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, dropDups: true})`	**Create a unique index on** *username*, **dropping any existing duplicates.**
`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: ‘2d’})`	**Create a geospatial index on** *loc.*

Administration

`db.users.getIndexes()`	**Get a list of all indexes on the users collection.**
`db.system.indexes.find()`	**Directly query the collection containing index definitions for this db.**
`db.users.totalIndexSize()`	**Number of bytes allocated by indexes for** *users* **collection.**
`db.stats()`	**Includes total index size for current database.**
`db.users.reIndex()`	**Rebuild all indexes on this collection.**
`db.users.dropIndex(‘username’ )`	**Drop the index called** *username*. **Use** *getIndexes()* **to get the name of each index.**

Tips

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

```
find({username: "Jones" });
find({username: /^Jones/ });
find({username: "Jones", date: "2010-12-01"});
find({username: "Jones" }).sort({date: -1});
find({}).sort({username: 1, date: 1}).limit(100);
```

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