

# MongoDB®

## Notes for Professionals

### Chapter 2: CRUD Operation

#### Section 2.1: Create

##### db.people.insert((name: 'Tom', age: 28));

Or

```
db.people.save((name: 'Tom', age: 28));
```

The difference with `save` is that if the passed document contains an `_id` field, if a document `_id` it will be updated instead of being added as new.

Two new methods to insert documents into a collection in MongoDB 3.2.x:

- Use `insertOne` to insert only one record

```
db.people.insertOne((name: 'Tom', age: 28));
```

Use `insertMany` to insert multiple records:

```
db.people.insertMany((name: 'Tom', age: 28), (name: 'John', age: 29), (name: 'Jane', age: 25));
```

Note that `insert` is highlighted as deprecated in every official language driver since being that the shell methods actually lagged behind the other drivers in implementing applies for all other CRUD methods.

#### Section 2.2: Update

Update the entire object:

```
db.people.update((name: 'Tom', age: 29, name: 'Tom'))
```

```
// New in MongoDB 3.2  
db.people.updateOne((name: 'Tom', age: 29, name: 'Tom')) //will not document.
```

```
db.people.updateMany((name: 'Tom'), {age: 29, name: 'Tom'}) //will document.
```

Or just update a single field of a document. In this case age:

```
db.people.update((name: 'Tom'), {$set: {age: 29}})
```

You can also update multiple documents simultaneously by adding a third documents where the name equals Tom:

```
db.people.update((name: 'Tom'), {$set: {age: 28}}, {multi: true})
```

```
db.people.update((name: 'Tom'), {$set: {age: 28}}, {multi: true})
```

```
// New in MongoDB 3.2  
db.people.updateOne((name: 'Tom'), {$set: {age: 28}}) //will update all matching documents.
```

```
db.people.updateMany((name: 'Tom'), {$set: {age: 28}}) //will update all matching documents.
```

If a new field is coming for update, that field will be added to the document.

```
db.people.update((name: 'Tom'), {$set: {age: 28}, $addToSet: {newField: 'value'}})
```

```
db.people.updateMany((name: 'Tom'), {$set: {age: 28}, $addToSet: {newField: 'value'}})
```

```
db.people.updateOne((name: 'Tom'), {$set: {age: 28}, $addToSet: {newField: 'value'}})
```

```
db.people.updateMany((name: 'Tom'), {$set: {age: 28}, $addToSet: {newField: 'value'}})
```

```
db.people.updateOne((name: 'Tom'), {$set: {age: 28}, $addToSet: {newField: 'value'}})
```

```
db.people.updateMany((name: 'Tom'), {$set: {age: 28}, $addToSet: {newField: 'value'}})
```

```
db.people.updateOne((name: 'Tom'), {$set: {age: 28}, $addToSet: {newField: 'value'}})
```

```
db.people.updateMany((name: 'Tom'), {$set: {age: 28}, $addToSet: {newField: 'value'}})
```

```
db.people.updateOne((name: 'Tom'), {$set: {age: 28}, $addToSet: {newField: 'value'}})
```

```
db.people.updateMany((name: 'Tom'), {$set: {age: 28}, $addToSet: {newField: 'value'}})
```

```
db.people.updateOne((name: 'Tom'), {$set: {age: 28}, $addToSet: {newField: 'value'}})
```

```
db.people.updateMany((name: 'Tom'), {$set: {age: 28}, $addToSet: {newField: 'value'}})
```

```
db.people.updateOne((name: 'Tom'), {$set: {age: 28}, $addToSet: {newField: 'value'}})
```

### Chapter 8: Aggregation

**Parameter** Details  
**pipeline** an array sequence of data aggregation operations or stages  
**options** document(optional, available only if pipeline present as an array)

Aggregation operations process data records and return computed results. Aggregation operations group values from multiple documents together, and can perform a variety of operations on the grouped data to return a single result. MongoDB provides three ways to perform aggregation: the aggregation pipeline, the map-reduce function, and single purpose aggregation methods.

From Mongo manual <https://docs.mongodb.com/manual/aggregation/>

#### Section 8.1: Count

How do you get the number of Debit and Credit transactions? One way to do it is by using `count()` function as below:

```
> db.transactions.count({cr_dr: 'C'});
```

Or

```
> db.transactions.find({cr_dr: 'C'}).length();
```

But what if you do not know the possible values of `cr_dr` upfront. Here Aggregation framework comes to play. See the below Aggregate query.

```
> db.transactions.aggregate(  
  {  
    $group: {  
      _id: '$cr_dr', // group by type of transaction  
      // Add 1 for each document to the count for this type of transaction  
      count: { $sum: 1 }  
    }  
  })
```

And the result is

```
{ "_id": "C",  
  "count": 3 }  
{ "_id": "D",  
  "count": 5 }
```

#### Section 8.2: Sum

How to get the summation of amount? See the below aggregate query.

MongoDB Notes for Professionals

### Chapter 9: Indexes

#### Section 9.1: Index Creation Basics

See the below transactions collection:

```
> db.transactions.insert({ cr_dr: 'D', amount: 100, fee: 2 });  
> db.transactions.insert({ cr_dr: 'D', amount: 100, fee: 2 });  
> db.transactions.insert({ cr_dr: 'D', amount: 100, fee: 2 });  
> db.transactions.insert({ cr_dr: 'D', amount: 100, fee: 2 });  
> db.transactions.insert({ cr_dr: 'D', amount: 100, fee: 2 });  
> db.transactions.insert({ cr_dr: 'D', amount: 100, fee: 2 });
```

`getIndexes()` functions will show all the indexes available for a collection.

```
db.transactions.getIndexes();
```

Let see the output of above statement:

```
{ "v": 1,  
  "key": {  
    "_id": 1  
  },  
  "name": "_id_",  
  "ns": "database.transactions" }
```

There is already one index for transaction collection. This is because MongoDB creates a unique index on the `_id` field during the creation of a collection. The `_id` index prevents clients from inserting two documents with the same value for the `_id` field. You cannot drop this index on the `_id` field.

Now let's add an index for `cr_dr` field:

```
db.transactions.createIndex({ cr_dr: 1 });
```

The result of the index execution is as follows:

```
{ "createdCollectionAutomatically": false,  
  "partialFilterExpression": {},  
  "unique": true,  
  "name": "cr_dr_1" }
```

The `createdCollectionAutomatically` indicates if the operation created a collection, if a collection does not exist, MongoDB creates the collection as part of the indexing operation.

Let run `db.transactions.getIndexes()` again:

MongoDB Notes for Professionals

**60+ pages**  
of professional hints and tricks

# Contents

<b>About</b>	1
<b>Chapter 1: Getting started with MongoDB</b>	2
Section 1.1: Execution of a JavaScript file in MongoDB	2
Section 1.2: Making the output of find readable in shell	2
Section 1.3: Complementary Terms	3
Section 1.4: Installation	3
Section 1.5: Basic commands on mongo shell	6
Section 1.6: Hello World	6
<b>Chapter 2: CRUD Operation</b>	7
Section 2.1: Create	7
Section 2.2: Update	7
Section 2.3: Delete	8
Section 2.4: Read	8
Section 2.5: Update of embedded documents	9
Section 2.6: More update operators	10
Section 2.7: "multi" Parameter while updating multiple documents	10
<b>Chapter 3: Getting database information</b>	11
Section 3.1: List all collections in database	11
Section 3.2: List all databases	11
<b>Chapter 4: Querying for Data (Getting Started)</b>	12
Section 4.1: Find()	12
Section 4.2: FindOne()	12
Section 4.3: limit, skip, sort and count the results of the find() method	12
Section 4.4: Query Document - Using AND, OR and IN Conditions	14
Section 4.5: find() method with Projection	16
Section 4.6: Find() method with Projection	16
<b>Chapter 5: Update Operators</b>	18
Section 5.1: \$set operator to update specified field(s) in document(s)	18
<b>Chapter 6: Upserts and Inserts</b>	20
Section 6.1: Insert a document	20
<b>Chapter 7: Collections</b>	21
Section 7.1: Create a Collection	21
Section 7.2: Drop Collection	22
<b>Chapter 8: Aggregation</b>	23
Section 8.1: Count	23
Section 8.2: Sum	23
Section 8.3: Average	24
Section 8.4: Operations with arrays	25
Section 8.5: Aggregate query examples useful for work and learning	25
Section 8.6: Match	29
Section 8.7: Get sample data	30
Section 8.8: Remove docs that have a duplicate field in a collection (dedupe)	30
Section 8.9: Left Outer Join with aggregation ( \$Lookup)	30
Section 8.10: Server Aggregation	31
Section 8.11: Aggregation in a Server Method	31
Section 8.12: Java and Spring example	32

<b>Chapter 9: Indexes</b>	34
Section 9.1: Index Creation Basics	34
Section 9.2: Dropping/Deleting an Index	36
Section 9.3: Sparse indexes and Partial indexes	36
Section 9.4: Get Indices of a Collection	37
Section 9.5: Compound	38
Section 9.6: Unique Index	38
Section 9.7: Single field	38
Section 9.8: Delete	38
Section 9.9: List	39
<b>Chapter 10: Bulk Operations</b>	40
Section 10.1: Converting a field to another type and updating the entire collection in Bulk	40
<b>Chapter 11: 2dsphere Index</b>	43
Section 11.1: Create a 2dsphere Index	43
<b>Chapter 12: Pluggable Storage Engines</b>	44
Section 12.1: WiredTiger	44
Section 12.2: MMAP	44
Section 12.3: In-memory	44
Section 12.4: mongo-rocks	44
Section 12.5: Fusion-io	44
Section 12.6: TokuMX	45
<b>Chapter 13: Java Driver</b>	46
Section 13.1: Fetch Collection data with condition	46
Section 13.2: Create a database user	46
Section 13.3: Create a tailable cursor	46
<b>Chapter 14: Python Driver</b>	48
Section 14.1: Connect to MongoDB using pymongo	48
Section 14.2: PyMongo queries	48
Section 14.3: Update all documents in a collection using PyMongo	49
<b>Chapter 15: Mongo as Shards</b>	50
Section 15.1: Sharding Environment Setup	50
<b>Chapter 16: Replication</b>	51
Section 16.1: Basic configuration with three nodes	51
<b>Chapter 17: Mongo as a Replica Set</b>	53
Section 17.1: Mongodb as a Replica Set	53
Section 17.2: Check MongoDB Replica Set states	54
<b>Chapter 18: MongoDB - Configure a ReplicaSet to support TLS/SSL</b>	56
Section 18.1: How to configure a ReplicaSet to support TLS/SSL?	56
Section 18.2: How to connect your Client (Mongo Shell) to a ReplicaSet?	58
<b>Chapter 19: Authentication Mechanisms in MongoDB</b>	60
Section 19.1: Authentication Mechanisms	60
<b>Chapter 20: MongoDB Authorization Model</b>	61
Section 20.1: Build-in Roles	61
<b>Chapter 21: Configuration</b>	62
Section 21.1: Starting mongo with a specific config file	63
<b>Chapter 22: Backing up and Restoring Data</b>	64
Section 22.1: Basic mongodump of local default mongod instance	64
Section 22.2: Basic mongorestore of local default mongod dump	64

<a href="#">Section 22.3: mongoimport with JSON</a>	64
<a href="#">Section 22.4: mongoimport with CSV</a>	65
<b><a href="#">Chapter 23: Upgrading MongoDB version</a></b>	66
<a href="#">Section 23.1: Upgrading to 3.4 on Ubuntu 16.04 using apt</a>	66
<b><a href="#">Credits</a></b>	67
<b><a href="#">You may also like</a></b>	69

# About

Please feel free to share this PDF with anyone for free,  
latest version of this book can be downloaded from:

<https://goalkicker.com/MongoDBBook>

This *MongoDB® Notes for Professionals* book is compiled from [Stack Overflow Documentation](#), the content is written by the beautiful people at Stack Overflow. Text content is released under Creative Commons BY-SA, see credits at the end of this book whom contributed to the various chapters. Images may be copyright of their respective owners unless otherwise specified

This is an unofficial free book created for educational purposes and is not affiliated with official MongoDB® group(s) or company(s) nor Stack Overflow. All trademarks and registered trademarks are the property of their respective company owners

The information presented in this book is not guaranteed to be correct nor accurate, use at your own risk

Please send feedback and corrections to [web@petercv.com](mailto:web@petercv.com)

[Click here to download full PDF material](#)