Introduction to MongoDB

Poo Kuan Hoong

Introduction to CRUD

- MongoDB stores data in the form of documents, which are JavaScript Object Notation (JSON) like field and value pairs
- JSON Syntax Example:

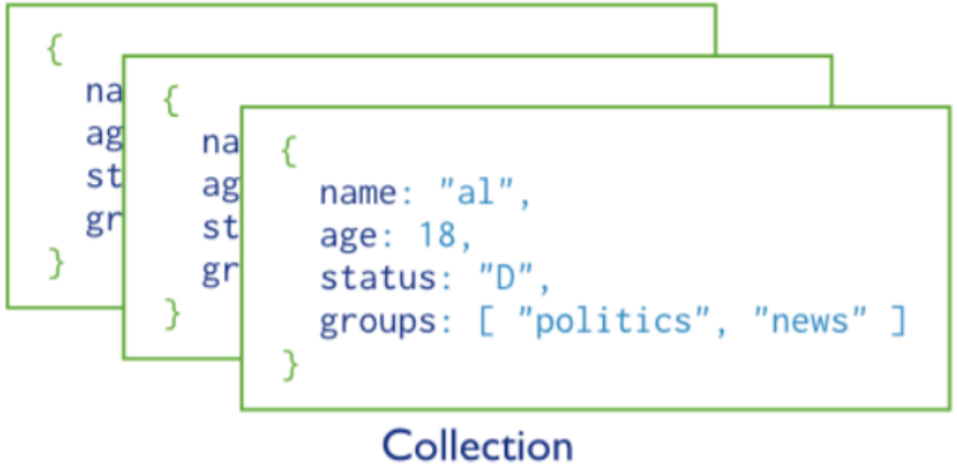
```
{ application: "HR System",
   user: [
   {name: "Kate",
    age: 35
   },
   {name: "Jack",
    age: 42
   }
]}
```

Value

Field

Introduction to CRUD

- MongoDB stores all documents in collections.
- A collection is a group of related documents that have a set of shared common indexes.
- Collections are analogous to a table in relational databases.



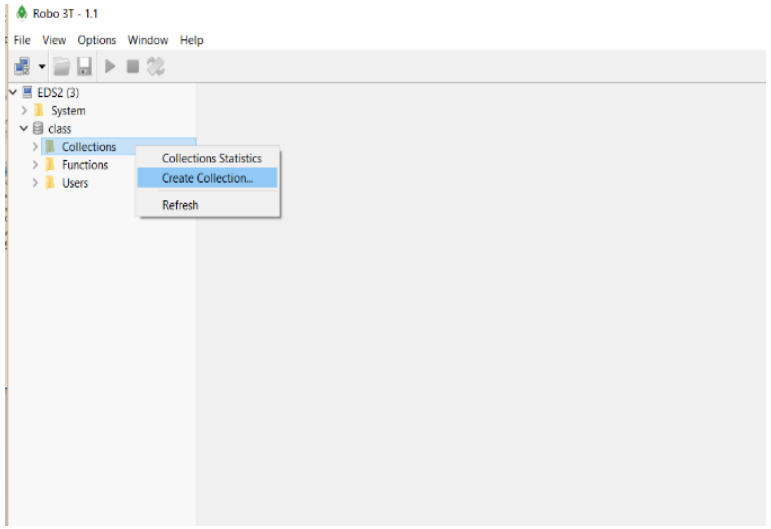
Introduction to CRUD

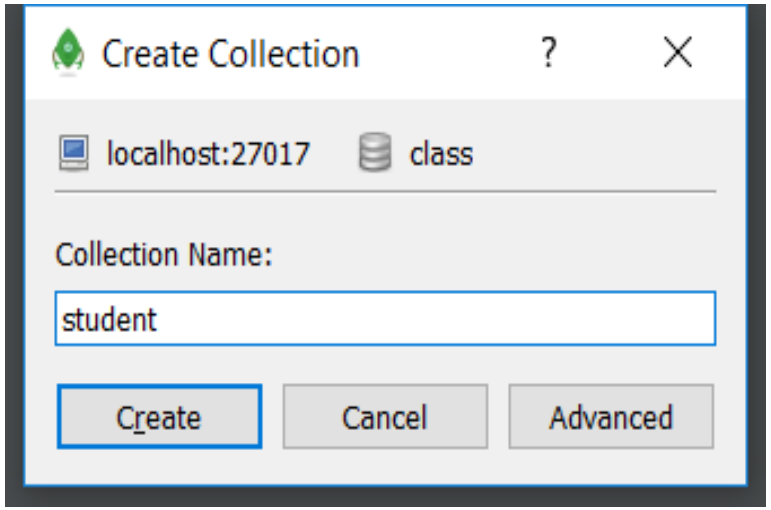
C	• insert ()
R	• find ()
U	• update ()
D	• remove ()

Creating Collection

- We have already created our database now it's time to create our collection.
- MongoDB will automatically create collection if it's not been created. Else, MongoDB will insert the record in the existing collection.
- Methods:

1. Expand the database and right click on Collections. Then, choose Create Collection.
2. Name the collection as 'Student' and click 'Create'





MongoDB

- MongoDB

```
db.users.insert (        ← collection
   {
     name: "sue",        ← field: value
     age: 26,            ← field: value
     status: "A"         ← field: value
   }
)
```

} document

- Relational Database

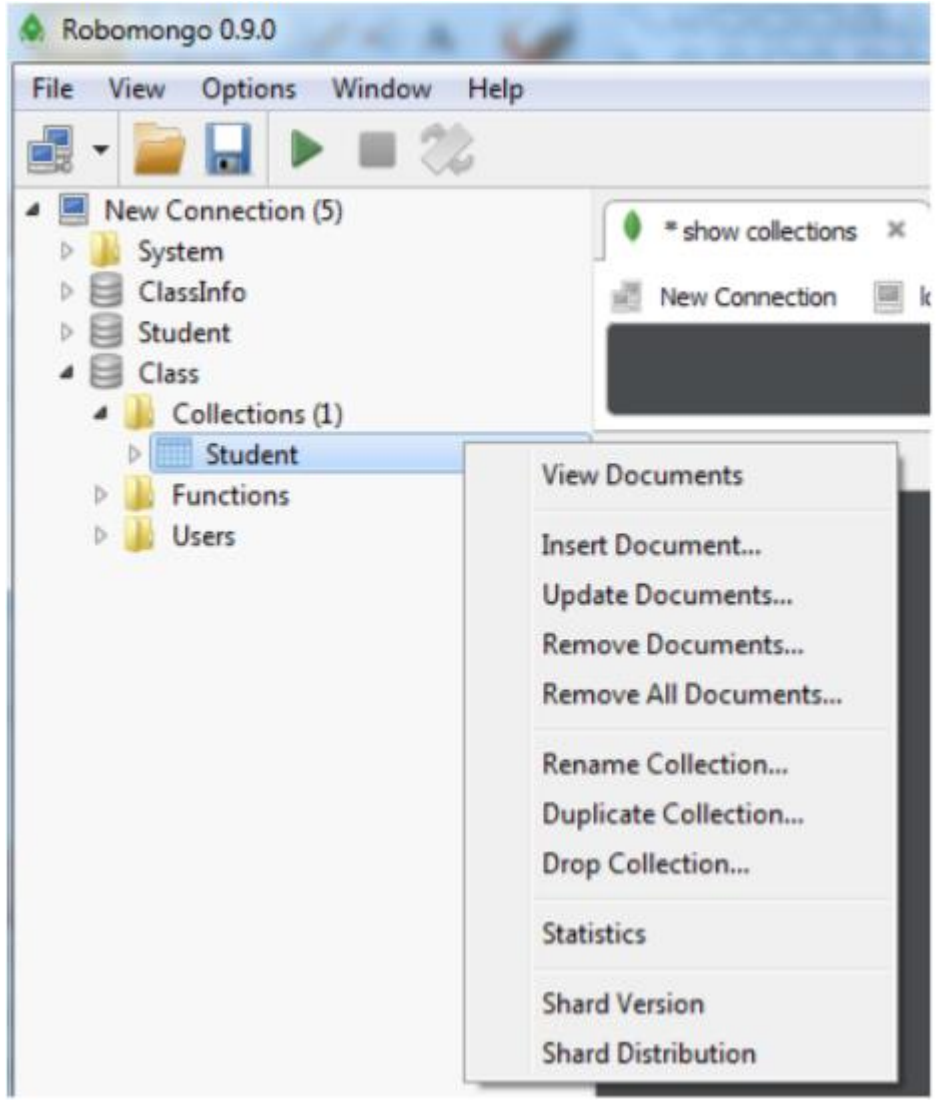
```
INSERT INTO users                    ← table
            ( name, age, status )    ← columns
VALUES      ( "sue", 26, "A" )       ← values/row
```

ObjectID

- We do not need to provide `_id` for any documents but MongoDB automatically create unique `_id` for each document inserted.
  - `"_id":ObjectId("56c2f6119d1f4f37a06823fc")`
- When we insert a document in MongoDB, server requires that all documents is uniquely identified using an `_id`.
- The `_id` field is the primary key.
- `ObjectId` will be constructed using:
  - current time, identifier for the machine that constructing the `ObjectId`, the process id of the process

Inserting Document

1. Expand ‘Collections’
2. Right click on ‘Student’
3. Choose ‘Insert Document’



Inserting Document (cont'd)

4. Type in the code within the '{}'
5. Click 'Validate' to check the syntax of the query
6. Click 'Save

CRUD Operations

C	• insert ()	√
R	• find ()	
U	• update ()	
D	• remove ()	

Finding Document

- In SQL, we use `SELECT` to query the database.
- In MongoDB, we use `find()`

SQL	MongoDB
`SELECT *` `FROM Student;`	`db.Student.find()`

Finding Document (cont'd)

```
db.Student.find()
```

Student 0 sec. 0 50

Key	Value	Type
(1) ObjectId("58afbcb24d20122597d3cd29")	{ 5 fields }	Object
_id	ObjectId("58afbcb24d20122597d3cd29")	ObjectId
StudID	A101	String
StudName	Kanesan	String
Quiz	[3 elements]	Array
[0]	21	Int32
[1]	19	Int32
[2]	20	Int32
Final	65	Int32
(2) ObjectId("58afbcb24d20122597d3cd2b")	{ 5 fields }	Object
_id	ObjectId("58afbcb24d20122597d3cd2b")	ObjectId
StudID	A102	String
StudName	Chong King Wai	String
Quiz	[3 elements]	Array
[0]	31	Int32
[1]	19	Int32
[2]	10	Int32
Final	75	Int32
(3) ObjectId("58afbcb24d20122597d3cd2d")	{ 5 fields }	Object
_id	ObjectId("58afbcb24d20122597d3cd2d")	ObjectId
StudID	A103	String
StudName	Catherine Ray	String
Quiz	[3 elements]	Array
[0]	0	Int32
[1]	7	Int32
[2]	10	Int32
Final	41	Int32
(4) ObjectId("58afbcb24d20122597d3cd2f")	{ 5 fields }	Object
_id	ObjectId("58afbcb24d20122597d3cd2f")	ObjectId
StudID	A104	String
StudName	Thomas	String
Quiz	[3 elements]	Array
[0]	20	Int32
[1]	20	Int32
[2]	30	Int32
Final	80	Int32

Finding Document (cont'd)

- Limit the search results using equality condition

SQL	MongoDB

`SELECT *`
`FROM Student`
`WHERE StudName = "Thomas";`
	`db.Student.find({StudName: "Thomas"})`

```
db.Student.find({StudName: "Thomas"})
```

Student 0.034 sec. 0 50

Key	Value	Type
(1) ObjectId("58afbcb24d20122597d3cd2f")	{ 5 fields }	Object
_id	ObjectId("58afbcb24d20122597d3cd2f")	ObjectId
StudID	A104	String
StudName	Thomas	String
Quiz	[3 elements]	Array
[0]	20	Int32
[1]	20	Int32
[2]	30	Int32
Final	80	Int32

Finding Document (cont'd)

- Finding documents with logical query operators - AND and OR

```
db.Student.find({$and:[{StudName: "Thomas"},
{StudID: "A104"}]})

db.Student.find({$or:[{StudName: "Thomas"},
StudName: "Kanesan"}]})
```

Finding Document (cont'd)

- Finding documents with comparison query operators:
  - Greater than `$gt`
  - Greater than equal `$gte`
  - Less than `$lt`
  - Less than equal `$lte`
  - Not equal `$ne`

```
db.Student.find({Final: {$gt: 50}})
```

? Activity

- Let’s examine the following commands. For each command,
  - Explain the purpose of the query
  - Execute the command in RoboMongo. How many documents are returned?

- `db.Student.find({Final: {$gte: 50}})`
- `db.Student.find({Final: {$lt: 50}})`
- `db.Student.find({Final: {$ne: 50}})`
- `db.Student.find({$and: [{Final: {$gte: 70}},{Final: {$lt: 80}}]})`
- `db.Student.find({StudName: {$ne: "Thomas"}})`
- `db.Student.find({$or: [{Final: {$lt: 50}}, {StudName: {$ne: "Thomas"}}] })`

Finding Document (cont'd)

- We can use IN for checking values of the same field

- `db.Student.find({StudName: {`**`$in:`**` ["Thomas", "Kanesan"]}})`
- Use aggregate() to perform aggregation operations:
  - `$sum` returns the summation value
  - `$avg` returns the average value
  - `$max` returns the maximum value
  - `$min` returns the minimum value

Finding Document (cont'd)

- Get the highest final mark in all documents

```
db.Student.aggregate
      ([{$group: {_id: null,
            highestMark: {$max: "$Final"}}
}])
```

$group arrange identical data into groups

_id is compulsory field. Set this to null to calculate accumulated values for all the documents as a whole

```
db.Student.aggregate([{$group: {_id: null, highestMark: {$max: "$Final" }} }])
```

0.001 sec.

Key	Value	Type
(1) null	{ 2 fields }	Object
_id	null	Null
highestMark	80	Int32

Finding Document (cont'd)

- Get the average and minimum final mark in all documents

```
db.Student.aggregate
([{$group: {_id: null,
            AverageMark: {$avg: "$Final" },
            MinMark: {$min: "$Final" }}
}])
```

```
db.Student.aggregate([{$group: {_id: null, AverageMark: {$avg: "$Final" }, MinMark: {$min: "$Final" }} }])
```

0.002 sec.

Key	Value	Type
(1) null	{ 3 fields }	Object
_id	null	Null
AverageMark	65.25	Double
MinMark	41	Int32

Finding Document (cont'd)

- Count the number of documents (students) in a collection

```
db.Student.aggregate
([{$group: {_id : null,
                TotalCount: {$sum: 1}} }])
```

```
db.Student.aggregate([{$group: {_id : null, TotalCount: {$sum: 1}} }])
```

0.001 sec.

Key	Value	Type
(1) null	{ 2 fields }	Object
_id	null	Null
TotalCount	4.0	Double

Finding Document (cont'd)

- Get the name and total quiz score for each student

```
db.Student.aggregate
([{$project: {StudName: 1,
                quizTotal: {$sum: "$Quiz"}}}])
```

$project select some specific fields from a collection

```
db.Student.aggregate([{$project: { StudName: 1, quizTotal: {$sum: "$Quiz"}}}])
```

0.001 sec.

Key	Value	Type
(1) ObjectId("58afbcb24d20122597d3cd29")	{ 3 fields }	Object
_id	ObjectId("58afbcb24d20122597d3cd29")	ObjectId
StudName	Kanesan	String
quizTotal	60	Int32
(2) ObjectId("58afbcb24d20122597d3cd2b")	{ 3 fields }	Object
_id	ObjectId("58afbcb24d20122597d3cd2b")	ObjectId
StudName	Chong King Wai	String
quizTotal	60	Int32
(3) ObjectId("58afbcb24d20122597d3cd2d")	{ 3 fields }	Object
_id	ObjectId("58afbcb24d20122597d3cd2d")	ObjectId
StudName	Catherine Ray	String

Finding Document (cont'd)

- Calculate the total mark (final + quiz) *for each* student

```
db.Student.aggregate
([{$project: {TotalMark:
        {$sum: [{$sum: "$Quiz"}, "$Final"]}} }])
```

```
db.Student.aggregate([{$project: {TotalMark: {$sum: [{$sum: "$Quiz"}, "$Final"] }} }])
```

0.009 sec.

Key	Value	Type
(1) ObjectId("58afbcb24d20122597d3cd29")	{ 2 fields }	Object
_id	ObjectId("58afbcb24d20122597d3cd29")	ObjectId
TotalMark	125	Int32
(2) ObjectId("58afbcb24d20122597d3cd2b")	{ 2 fields }	Object
_id	ObjectId("58afbcb24d20122597d3cd2b")	ObjectId
TotalMark	135	Int32
(3) ObjectId("58afbcb24d20122597d3cd2d")	{ 2 fields }	Object
_id	ObjectId("58afbcb24d20122597d3cd2d")	ObjectId
TotalMark	58	Int32
(4) ObjectId("58afbcb24d20122597d3cd2f")	{ 2 fields }	Object

Finding Document (cont'd)

- Count the number of quizzes taken by each student

```
db.Student.aggregate
([{$project: {QuizzesTaken:
                {$size: "$Quiz"}}}])
```

```
db.Student.aggregate([{$project: {QuizzesTaken: {$size: "$Quiz"}}}])
```

0.001 sec.

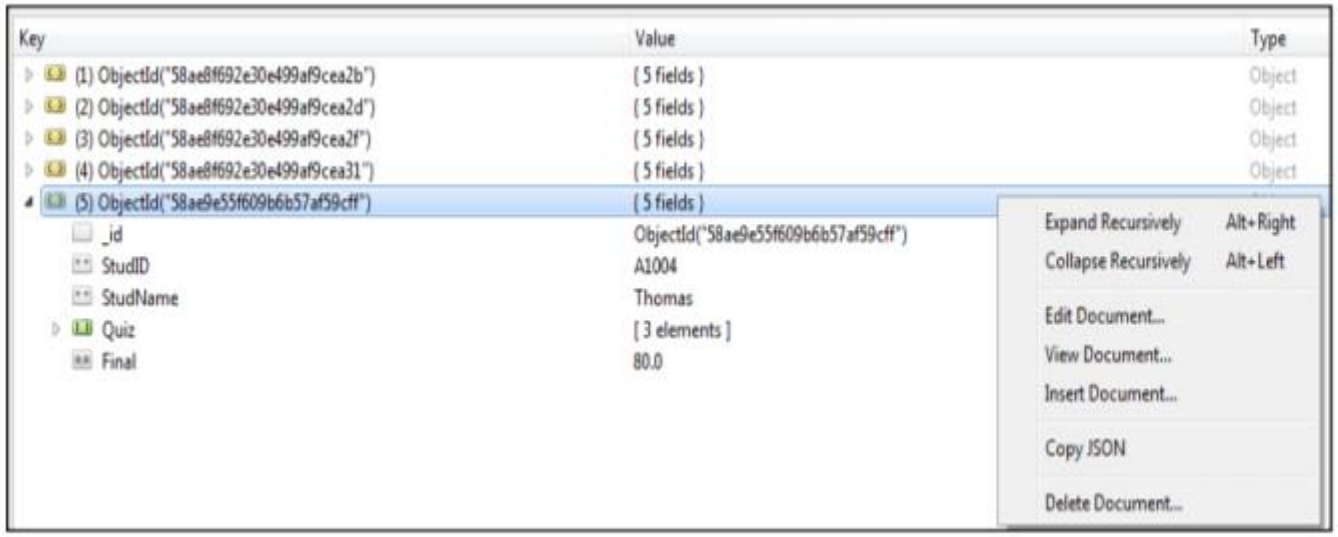
Key	Value	Type
(1) ObjectId("58afbcb24d20122597d3cd29")	{ 2 fields }	Object
_id	ObjectId("58afbcb24d20122597d3cd29")	ObjectId
QuizzesTaken	3	Int32
(2) ObjectId("58afbcb24d20122597d3cd2b")	{ 2 fields }	Object
_id	ObjectId("58afbcb24d20122597d3cd2b")	ObjectId
QuizzesTaken	3	Int32
(3) ObjectId("58afbcb24d20122597d3cd2d")	{ 2 fields }	Object
_id	ObjectId("58afbcb24d20122597d3cd2d")	ObjectId
QuizzesTaken	3	Int32
(4) ObjectId("58afbcb24d20122597d3cd2f")	{ 2 fields }	Object
_id	ObjectId("58afbcb24d20122597d3cd2f")	ObjectId

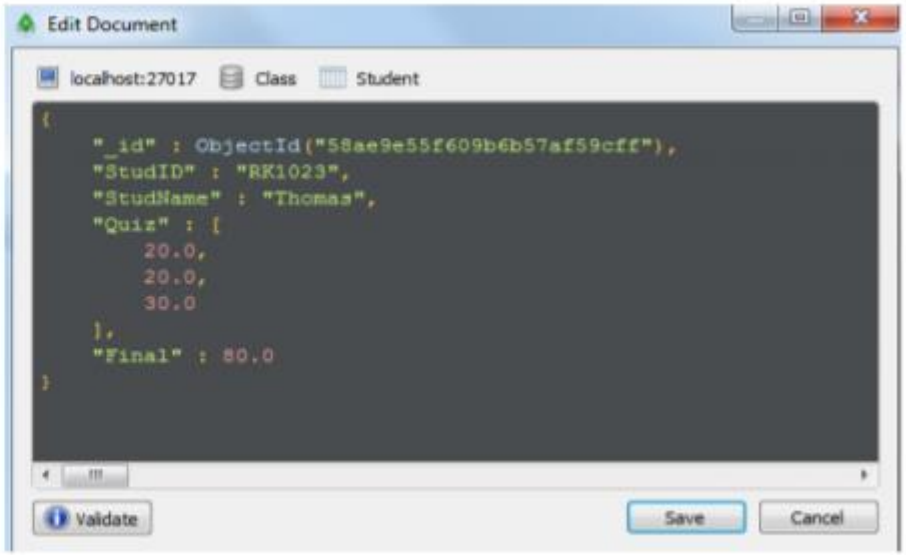
CRUD Operations

C	• insert ()	√
R	• find ()	√
U	• update ()	
D	• remove ()	

Updating document

1. Right click on the document to edit
2. Choose ‘Edit Document’
3. Edit the script with new value(s)
4. Click ‘Validate’ followed by ‘Save’



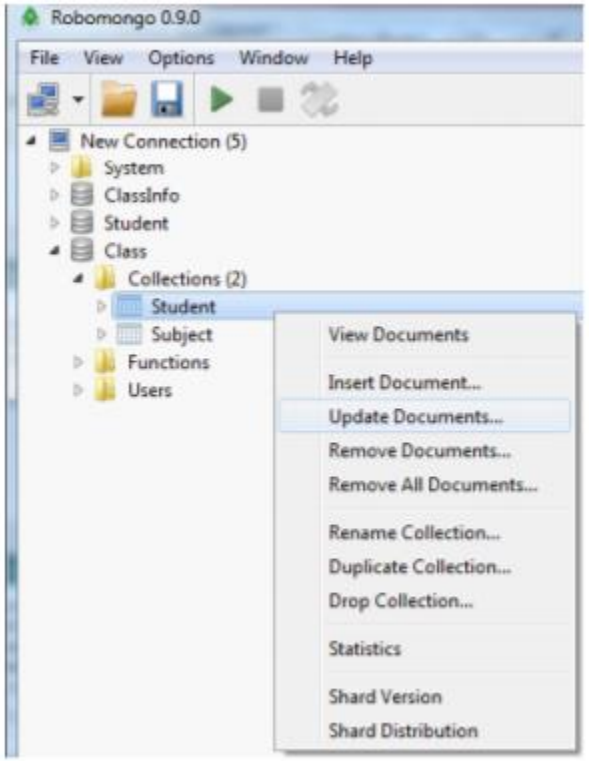


Updating document (Cont'd)

- The `update()` method is used to update the documents of a collection.
- The method accepts parameters:
  - An update conditions document to match the documents to update,
  - An update operations document to specify the modification to perform, and
  - Options parameters.
- By default, `update()` updates a single document. To update multiple documents, use the multi option.

Updating document (Cont'd)

1. Right click on the document to update
2. Choose 'Update Documents'
3. Edit the script with conditions and new value(s)
4. Click 'Validate' followed by 'Save'




```
db.getCollection('Student').update(
    // query
    {
        StudName: "Thomas"
    }
    ,
{$set: {StudName: "Thomas Mathew"}},
// options
    {
        "multi" : false,  // update only one document
        "upsert" : false  // insert a new document, if no existing document match the query
    }
);
```

Condition

New value assigned to StudName

Updating document (Cont'd)

- The method to add a new field into a collection is same as the previous process. If the field in the update query is not found in the document, then field will be added to the document

```
db.getCollection('Student').update(
    // query
    {
      StudName:"Chong King Wai"
    },

    // update
    {
        $set:{
            Course: "BIT"
        }
    },

    // options
    {
        "multi" : true,
        "upsert" : false
    }
```

Course field will be added to the document

Updating document (Cont'd)

- To delete a particular field in a collection: **$unset**
- s

```
db.getCollection('Student').update(
    // query
    {
        StudName:"Thomas Mathew"
    },

    // update
    {
        $unset:
        {
            Quiz: ""
        }
    },

    // options
    {
        "multi" : false,
        "upsert" : false
    }
);
```

'Quiz' field will be deleted from Thomas Mathew's document

CRUD Operations

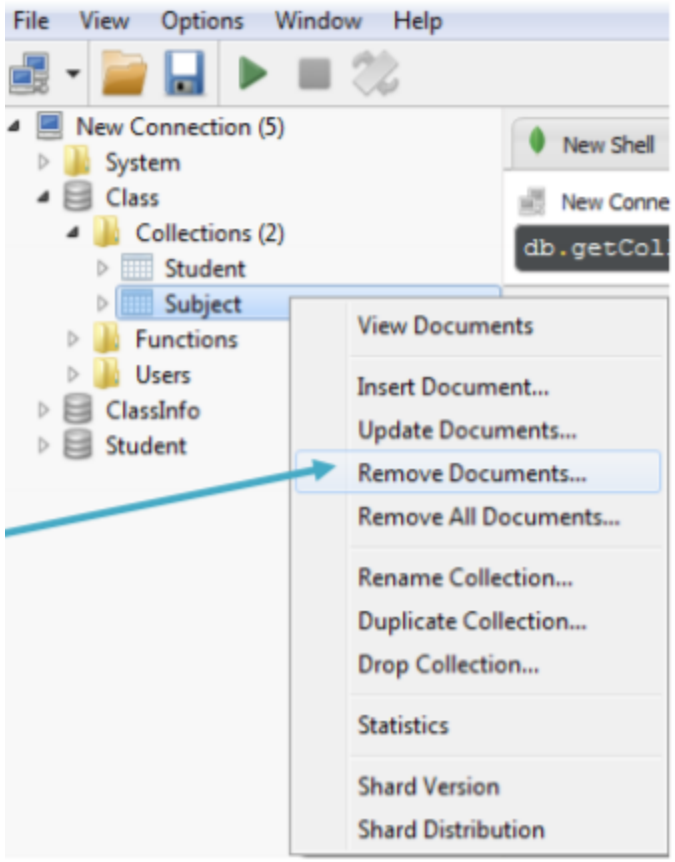
C	• insert ()	√
R	• find ()	√
U	• update ()	√
D	• remove ()	

Remove Documents

- Use remove() to remove a document from a collection that satisfies user's condition
- Method

1. Right-click on the collection.
2. Choose 'Remove Documents'.
3. Execute the following script to remove documents where the subject code is 'DPF555'

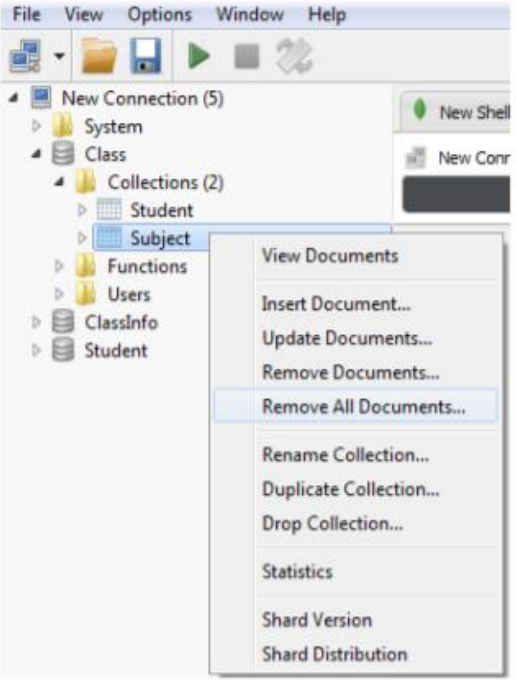
```
db.getCollection('Subject').
remove({SubjectCode :
"DPF555"});
```

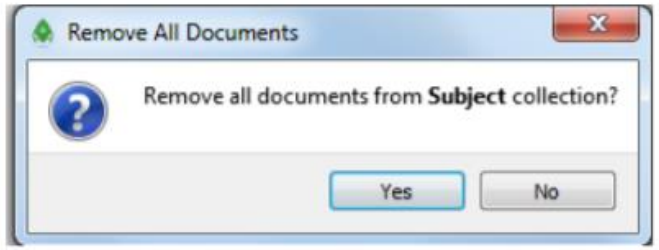


Remove Documents (Cont'd)

- To remove all documents:

1. Right-click on the collection
2. Choose 'Remove All Documents'
3. Click 'Yes' to remove all documents from the collection



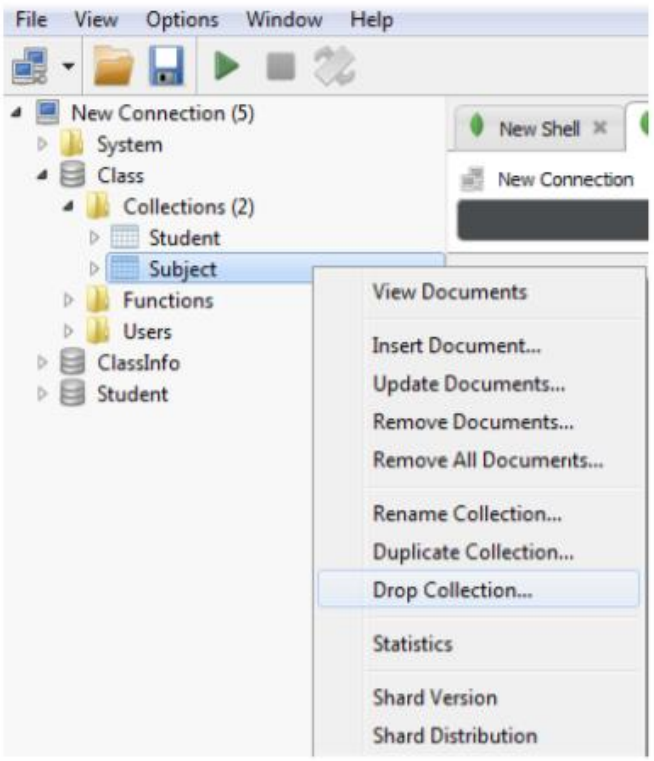


CRUD Operations

C	• insert ()	√
R	• find ()	√
U	• update ()	√
D	• remove ()	√

Drop a Collection from Database

1. Right-click on the collection
2. Choose ‘Drop Collection’
3. Click ‘Yes’ to drop the collection



Drop a database

1. .Right-click on the database
2. Choose ‘Drop Database’
3. Click ‘Yes’ to drop the database