MongoDB

Prof. Rachana V. Modi

Introduction of MongoDB

- NoSQL database
- Open Source
- Works on concept of collection and document

Database:

- Database is a physical container for collections.
- Each database gets its own set of files on the file system.
- A single MongoDB server typically has multiple databases.

Collection:

- Collection is a group of MongoDB documents.
- It is the equivalent of an RDBMS table.
- A collection exists within a single database.
- Collections do not enforce a schema.
- Documents within a collection can have different fields.

Introduction of MongoDB

Document

- A document is a set of key-value pairs.
- Documents have dynamic schema.
- Dynamic schema means documents in the same collection do not need to have the same set of fields or structure and common fields in a collection's documents may hold different types of data.

RELATIONSHIP OF RDBMS TERMINOLOGY WITH MONGODB

RDBMS	MongoDB
Database	Database
Table	Collection
Tuple/Row	Document
Column	Field
Table Join	Embedded Documents
Primary Key	Primary Key (Default key _id provided by MongoDB itself)
Database Server and Client	
mysqld/Oracle	Mongod
mysql/sqlplus	Mongo

Working of MongoDB

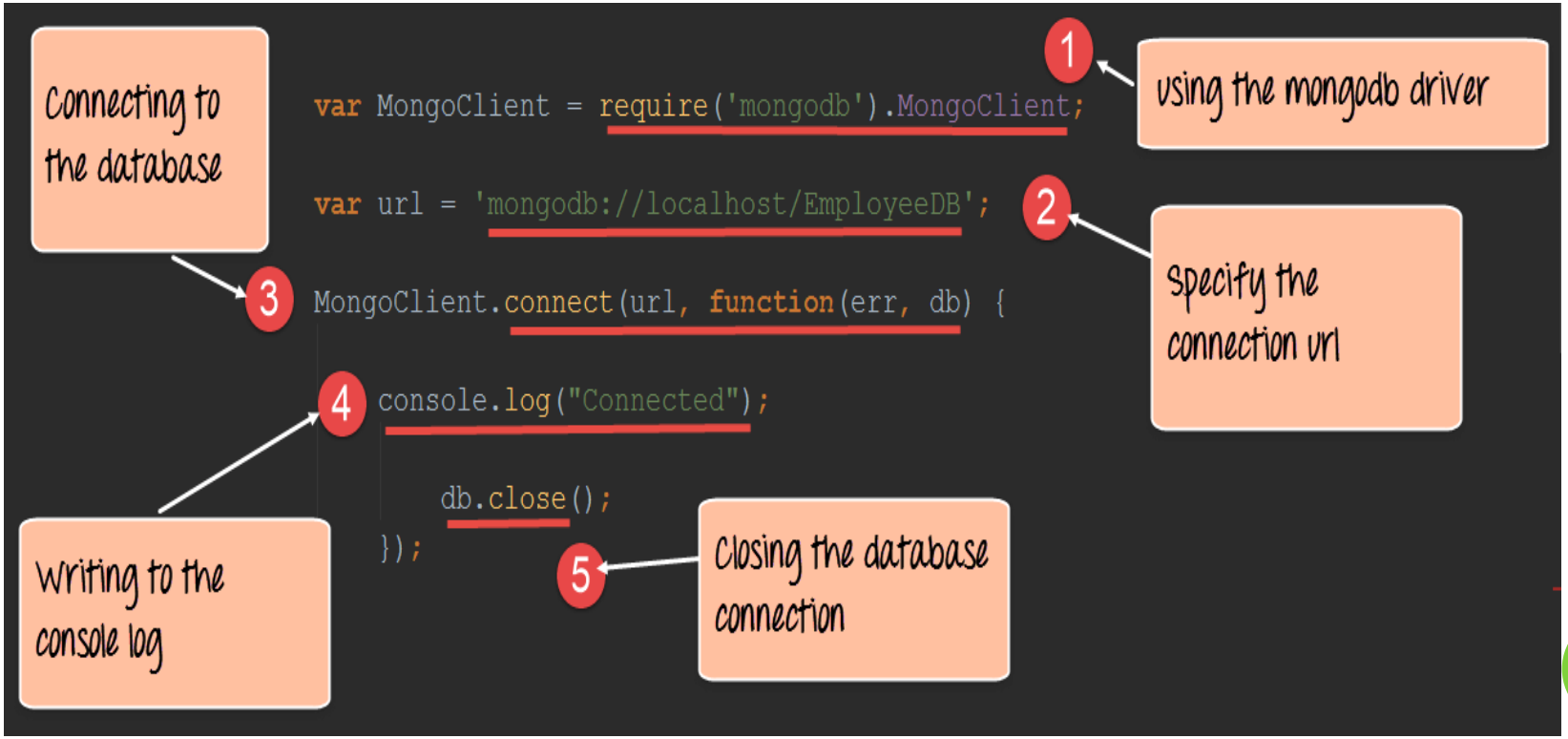
- Install MongoDB database in your system
- MongoDB requires a data folder to store its files. The default location for the **MongoDB data directory is c:\data\db**
- Start mongod (Mongo Daemon). mongod is a background process used by MongoDB. The main purpose of mongod is to manage all the MongoDB server tasks.

  For example accepting requests, responding to client and memory management.

  To start mongod write in command prompt: **C:\Program Files\MongoDB\Server\3.2\bin>mongod.exe**
- Run the MongoDB, you need to open another command prompt and issue the following command. **C:\Program Files\MongoDB\Server\3.2\bin>mongo.exe**

Create Application

- Install MongoDB Driver:**npm install mongodb**



Create Collection

- A collection in MongoDB is the same as a table in MySQL.
- To create a collection in MongoDB, use the createCollection() method and Create a collection of "students":

```
var MongoClient = require('mongodb').MongoClient;
var url="mongodb://localhost/";

MongoClient.connect(url, function(err, db) {
  if (err) throw err;
  var dbo = db.db("class_demo");
  dbo.createCollection("students", function(err, res) {
    if (err) throw err;
    console.log("Collection created!");
    db.close();
  });  });
```

Insert Documents

- insertOne() method is used to insert document in collection.
- The first parameter of the insertOne() method is an object containing pair of the name(s) and value(s) of each field in the document.
- Second parameter takes a callback function where you can work with any errors, or the result of the insertion.

Insert Documents

Example: Insert a document in the "students" collection

```
var MongoClient = require('mongodb').MongoClient;
var url = "mongodb://localhost/";

MongoClient.connect(url, function(err, db) {
  if (err) throw err;
  var dbo = db.db("class_demo");
  var obj = { name: "Alpesh", standard: 10 };
  dbo.collection("students").insertOne(obj, function(err, res) {
    if (err) throw err;
    console.log("1 document inserted");
    db.close();
  });
});
```

Insert Documents

Example: Insert many documents in the "students" collection

```
var MongoClient = require('mongodb').MongoClient;
var url = "mongodb://localhost/";

MongoClient.connect(url, function(err, db) {
  if (err) throw err;
  var dbo = db.db("class_demo");
   var obj = [
    { name: 'Juhi', standard: 1},
    { name: 'Palak', standard: 3 },
    { name: 'Anny', standard: 5},
    { name: 'Hitansh', standard: 11}
  ];
  dbo.collection("students").insertMany(obj, function(err, res) {
    if (err) throw err;
    console.log("res.insertedCount+" documents inserted");
    db.close();
  });   });
```

Select Document

- find() and findOne() methods are used to find document(s) in a collection.

Find One():

- The findOne() method returns the first occurrence in the selection.
- The first parameter of the findOne() method is a query object.
- Here we use an empty query object, which selects all documents in a collection (but returns only the first document).

Select Document

Example: Find the first document in the students collection

```
var MongoClient = require('mongodb').MongoClient;
var url = "mongodb://localhost/";

MongoClient.connect(url, function(err, db) {
  if (err) throw err;
var dbo = db.db("class_demo");
  dbo.collection("students").findOne({}, function(err, result) {
    if (err) throw err;
    console.log(result.name);
    db.close();
  });
});
```

Select Documents

Find():

- To select data from documents in MongoDB, we can also use the find() method.
- The find() method returns all occurrences in the selection.
- The first parameter of the find() method is a query object.
- In example we use an empty query object, which selects all documents in the collection.

Note: No parameters in the find() method gives you the same result as SELECT * in MySQL.

Select Documents

Example: Find all document in the students collection

```
var MongoClient = require('mongodb').MongoClient;
var url = "mongodb://localhost/";

MongoClient.connect(url, function(err, db) {
  if (err) throw err;
var dbo = db.db("class_demo");
  dbo.collection("students").find({}).toArray(function(err, result) {
    if (err) throw err;
    console.log(result);
  });
});
```

Delete Document

- To delete a document as it is called in MongoDB, we use the deleteOne() method.
- The first parameter of the deleteOne() method is a query object defining which document to delete.
- Note: If the query finds more than one document, only the first occurrence is deleted.

Delete Document

Example: Delete the document with the name “Juhi"

```
var MongoClient = require('mongodb').MongoClient;
var url = "mongodb://localhost/";

MongoClient.connect(url, function(err, db) {
  if (err) throw err;
 var myquery = { name: 'Juhi' };
  dbo.collection("students").deleteOne(myquery, function(err,
    obj) {
    if (err) throw err;
    console.log("1 document deleted");
});     });
```

Delete Documents

- deleteMany() method is used to delete many documents from a collection.
- The first parameter of the deleteMany() method is a query object defining which documents to delete.
- Result object returns an object which contains information about how the execution affected the database.

Delete Documents

Example: Delete all documents whose name starts with the letter "Ju":

```
var MongoClient = require('mongodb').MongoClient;
var url = "mongodb://localhost/";

MongoClient.connect(url, function(err, db) {
  if (err) throw err;
  var myquery = { name: /^Ju/ };
  dbo.collection("students").deleteMany(myquery, function(err, obj) {
    if (err) throw err;
    console.log(obj.result.n + " document(s) deleted");   //Return the number
     of deleted documents:
    db.close();
});     });
```

Update Document(s)

- **updateOne() and updateMany() methods are useful for updating documents**.
- The first parameter is a query object defines which document you want to update.
- The second parameter is an object defining the new values of the document.
- Note: If the query finds more than one record, only the first occurrence is updated via updateOne() method.
- The updateOne() and the updateMany() methods return an object which contains information about how the execution affected the database.

Update Document(s)

Example: Update the document with the name="Palak" and standard=4 when standard=3

```
var MongoClient = require('mongodb').MongoClient;
var url = "mongodb://localhost/";

MongoClient.connect(url, function(err, db) {
  if (err) throw err;
   var dbo = db.db("class_demo");
   var q1 = { standard: 3 };
  var newvalues = { $set: {name: "Palak", standard: 4 } };  // By using $set operator, only the specified fields are updated.
  dbo.collection("students").updateOne(q1, newvalues, function(err, res) {
    if (err) throw err;
    console.log("1 document updated");
    db.close();});     });
```

Update Document(s)

Example: Update all documents where the standard is 11

```
var MongoClient = require('mongodb').MongoClient;
var url = "mongodb://localhost/";

MongoClient.connect(url, function(err, db) {
  if (err) throw err;
  var dbo = db.db("class_demo");
  var q1 = { standard: 11 };
  var newvalues = { $set: {standard: 12 } };
  dbo.collection("students").updateMany(q1, newvalues, function(err, res) {
    if (err) throw err;
    console.log(res.result.nModified + " document(s) updated");
                          //Return the number of updated documents:

    db.close();
  });     });
```

Any query??