MongoDB: popular database

It can be used in mobile apps, desktop apps or only in the backend.

Mostly MongoDB is used in full stack web applications.

Express is a web server for Node.js

MongoDB uses Mozilla’s SpiderMonkey Javascript engine.

**Lecture 351**:   
Relational DB, Document DB (NoSQL)

In document database, data is stored in different documents and each document is different from others.

And it is not mandatory to have the schema for all documents.

All information related to a document is stored in 1 document, in 1 place.

When we perform a query, we just retrieve 1 document. No need to perform different join requests.

We can edit a document independently from others, insert a new document, delete a document, etc.

**Lecture 351**:

Each mongoDB database consists of databases.

There are default databases that are created after each installation of mongoDB and they are called ‘admin’ and ‘local’.

Each database consists of collections.

And each collection consists of documents.

Documents are grouped into the same collection by common fields.

It is not mandatory to have same fields for the documents in the same collection.

**Lecture 352**:

Mongo Shell, mongo

Mongo Server, mongod

When we install mongoDB, we get these 2 applications.

mongod is used to launch mongo server and mongo command is used to launch mongo shell.

These 2 applications are separate.

Mongo server stores data in the mongoDB database. It is the main point of data storage.

Mongo shell is used for the management of mongo server.

Using mongo shell, we can insert documents into the mongo server, etc.

We can perform all kinds of CRUD operations using mongo shell.

Both mongo shell and mongo server are based on the javascript engine.

In order to manage mongodb server, we can use mongo shell located on the same machine but we can also manage remote mongo server using local mongo shell.

**Lecture 353**:

Mongodb shell is installed along with mongodb server.

No username and password will be asked if mongodb server and mongo shell are on the same computer.

MongoDB Atlas

Useful when we want to use mongoDB in production.

We can also setup our own server and configure everything ourself.

mongoDB Atlas provides fully functional production solution for mongoDB.

We can connect with mongoDB atlas cluster using different ways.

We will be prompted to create an admin user when creating a new cluster.

It is recommended to use same versions of mongoDB shell and mongoDB server.

But for basic operations it does not matter.

**Lecture 356**: Exploring MongoDB Shell

Make sure that mongoDB server is running.

Mongo uses js engine.

db.help()

db.versions()

show dbs

3 databases are created.

“admin”, “config” and “local”

use admin

switched to db admin

Here we can list all collections inside the admin database.

“show collections”

“use local”

Switched to db local

“show collections”

startup\_log

db.stats() returns statistics about mongoDB server.

**Lecture 357**: Robo3T

It is not convenient to use native mongoDB shell in the terminal. All outputs and commands are in the same colour.

Robo3T is a lightweight GUI for MongoDB enthusiasts.

**Lecture 358**:

JSON

BSON

Extended JSON

JSON: Javascript Object Notation

6 data types:

String, Number, Object, Array, Boolean, Null

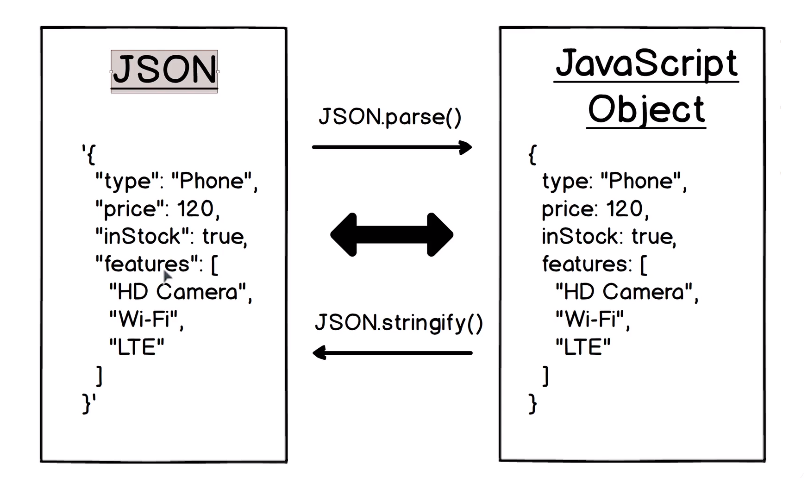
Quotes surround the JSON object. This means that the object is simply a string.

JSON object is a string and is used for data exchange between client and server or between different servers.

JSON is used for data exchange and is represented as a string.

JSON is a string.

Javascript object is a variable type.



Javascript object is a set of key-value pairs surrounded by curly braces.

In JSON, the key must be surrounded with double quotes.

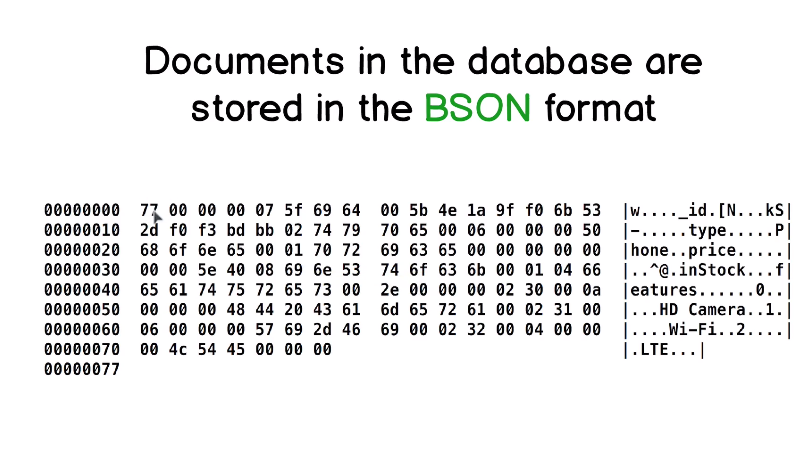
In javascript object, we can use double quotes around each key but it is not required.

We can convert JSON into Javascript object and back as well.

**Lecture 360**:

Documents in the mongoDB database are stored in the BSON format.

BSON is binary JSON.



This document is stored in BSON format and it is the previous document.

Document is stored in binary format and we can see representation of data in ascii codes and each ascii code is represted as pair of hexadecimal numbers.

Ascii Hex Symbol

Symbol w has hexadecimal value of 77.

Right part is an attempt to parse the middle part.

When there is no corresponding letter, number or special symbol for a specific ascii code, we will see a dot.

Why do we need this BSON format ?

Why we cannot use JSON to store data in the database ?

JSON supports only 6 data types.

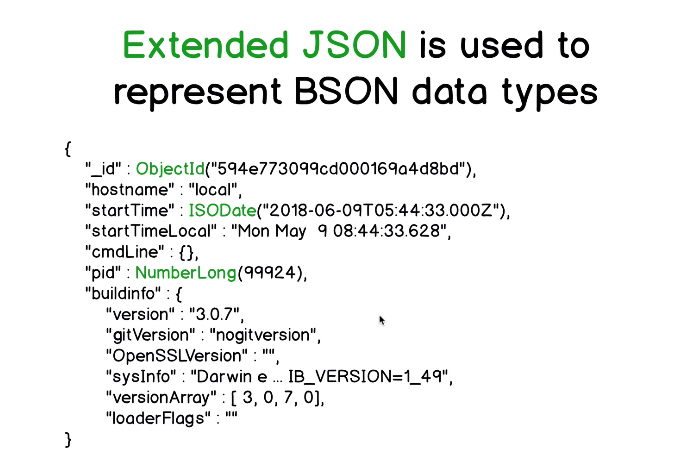
BSON supports many more data types.

The types are encoded.

Extended JSON is used to represent this set of characters in a readable format.

**Lecture 361**: Extended JSON

Extended JSON is used to represent BSON data types.



Extended JSON is used to represent data stored in BSON format in readable format.

It is also used to store new documents in the BSON format.

Using extended JSON, we can give instructions to mongoDB about which types should be used for values that we pass to the mongoDB.

In this example we can some values embedded in () and prefix such as ObjectId, ISODate and NumberLong are added.

This syntax combines JSON and javascript syntax.

ObjectId(“…”) looks like a call to the function.

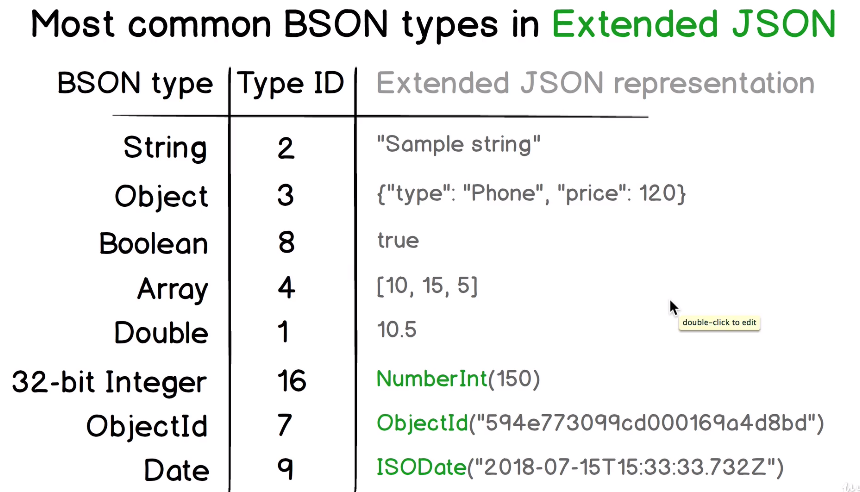
NumberLong(…)

This is one of the representation of a document from mongo database.

Extended JSON is something between JSON and BSON.

**Lecture 362**:

Most common BSON types in Extended JSON



String, Object, Boolean, Array, Double do not have any prefix because there is a corresponding value type in the JSON format.

There is no need to add additional information.

Any number in mongoDB if not instructed is stored as a double.

For 32 bit integer: NumberInt()

For 64 bit integer: NumberLong()

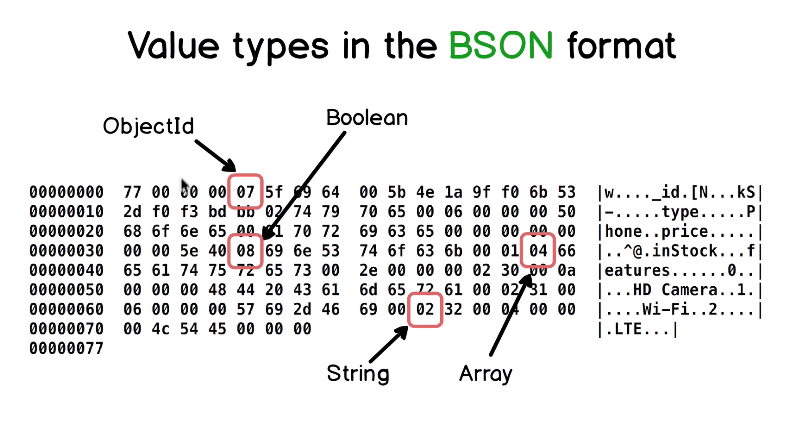
ObjectId is used in each mongoDB document and its goal is to make each document unique.

This number must be unique for each collection in mongoDB.

**Lecture 363**:

Value types in the BSON format

Type ids



There are value types encoded as separate characters.

07 code stores data type of \_id value.

08 is for boolean which is the value for inStock.

04 is for array. 02 is for string and we can see 3 02s representing 3 strings in the array.

Extended JSON is used for representation of BSON format into a readable format.

Extended JSON is also used to give mongoDB instructions, which value types it should use for data storage.

**Lecture 364**: Creation of new collections

Create collections and insert documents into it.

db.createCollection(“name”)

After installation of mongoDB server, we get 3 databases, local, admin and config.

We should not use these databases for our data.

Create a separate database.

“use forum”: this database won’t be created yet. This database will be created when we insert a new collection into it.

db.createCollection(“posts”)

“show collections”

posts

**Lecture 365**:

insertOne({…})

insertMany( [ {}, {}, …., {} ] )

These are collection methods.

insertOne inserts one document in the mongoDB collection.

Insert objects as documents in the collection.

We can easily use javascript objects in the mongoDB shell because mongoDB shell is based on Javascript.

After insertion, mongoDB server returns the object in extended JSON format.

db.posts.insertMany(…)

We can use the same ‘const’ variable name in a new mongo shell.

**Lecture 368**:

find({query})

findOne({query})

find returns all documents that match the query.

query is just a javascript object with key-value pairs.

find({})

Passing empty object will return all of the documents.

findOne({}) returns first document.

findOne({postId: ….})

// finds the document which matches the specified postId.

To access embedded property, use “”

find( { “author.name”: “abc” } )

If the document has

“tags”:[

“programming”,

“coding”

]

find( { tags: “programming” } )

This returns documents which have “programming” as one of the value in the “tags” array.

**Lecture 371**: Query operators

$or $eq $lt

$and $ne $gt

$in $nin $regex

db.getCollection(‘posts’).find( { comments:

{ $gt: 0 } } )

db.getCollection(‘posts’).find( {

$and: [

{ comments: { $lt: 5 } },

{ comments: { $gt: 0 } }

]

} )

db.getCollection(‘posts’).find( {

$or: [

{ shared: true },

{ tags: “programming” }

]

} )

$in operator also requires an array of elements as value.

db.getCollection(‘posts’).find({

tags: { $in: [ “programming”, “coding” ] }

})

**Lecture 373**: sort(), limit() and skip()

These are helper methods of the find() method.

Can use ascending or descending sort.

find({}).limit(2)

find({}).skip(2)

// first 2 documents will be skipped

find({}).sort({ comments: -1 })

// sort by comments in descending order.

comments: 1

// ascending order

By default, documents are sorted by \_id.

find({}).skip(2).sort({shared: 1})

false comes before true

**Lecture 374**: Update Methods

updateOne(), updateMany()

(<query>, <update>, <options>)

updateOne() just updates one document while updateMany() updates many documents simultaneously.

In <query>, we specify which documents we want to update.

In <update>, we specify what we want to update.

These 2 methods are collection methods.

We cannot use these methods without update operators.

**Update Operators**:

$set $rename

$unset $currentDate

$inc $addToSet

$set sets value of certain field.

$unset unsets certain field.

$inc increments certain field by specified amount.

**Lecture 376**: <query>, <update>, <options>

db.posts.updateOne(

{ postId: 2 },

{ $set: { shared: true } }

)

$unset deletes certain field.

db.posts.updateMany(

{ tags: [ ] },

{ $unset: { tags: 1 } }

)

**Lecture 378**: Deleting Documents

deleteOne({ query })

deleteMany({ query })

db.getCollection(‘posts’).deleteOne ({ postId: 12 })

deleteMany({ title: { $exists: false } })

**Lecture 379**: MongoDB Aggregation Framework

Using aggregation framework, we can take documents in our collection, we can find subset of the documents and produce brand new documents in the end.

Using aggregation framework, we can find all distinct values of the tags in the tags[ ] array in all documents.

db.posts.aggregate( [

{ $group: { \_id: “author.name” } }

] )

**Lecture 381**: MongoDB indexes

“use forum”

db.posts.getIndexes()

In each collection, there is a default index and this index is created by the field “\_id”.

{

“v”: 2,

“key”: {

“\_id” : 1

},

“name”: “\_id\_”

“ns”: “forum.posts”

}

1 means that the index is ascending.

Name of the index property.

Namespace is forum.posts

This index ensures that if we perform a query by \_id, for example

db.posts.findOne({ “\_id”: ObjectId(“…..”) })

The query will be executed by index.

Indexes are created for fields that are most often used in the queries.

**Lecture 382**: MongoDB utilities

mongoexport

mongoimport

mongodump

mongorestore

These utilities are separate applications and can be used for data management in mongoDB.

“mongoexport” is used to export data from certain collection.

“mongoimport” is used to import data into mongoDB database.

“mongodump” creates a dump of the entire mongoDB database.

“mongorestore” is used to restore data from the mongoDB dump.

To use these utilities, we do not need to open mongo shell.

“mongoexport -d forum -c posts -o posts.txt”

-d: name of database

-c: name of collection

-o: name of the file where the export data will be written to.

Parse command line arguments.

Apache Commons CLI

posts.txt is exported as a JSON object.

To open bson file, hexdump utility can be used.

hexdump abc.bson

hexdump -C abc.bson

(attempt to parse the sequence of ascii codes)

**Lecture 383**: MongoDB replica set

To use mongoDB in production, we must set up mongoDB replica sets.

Replica set consists of several mongoDB servers.

They have never seen production db that consists of just of 1 server.

In each replica set there is 1 primary server and all others are secondary.

We can make changes only on the primary server.

If we update one of the documents then this update is automatically propagated to all of the secondary servers.

In other words, it is replicated from primary to secondary. That is the process in the replica set.

If primary server fails, then secondary servers will elect new primary server and one of the secondary servers will become primary.

If we want to do write operation, then we must do it on primary server.

Read operation can be done on both primary and secondary servers.

mongoDB Atlas setup

We can see tags of primary and secondary under replica set.

MongoDB drivers

If we want to use mongoDB in full stack or back end applications, then we must consider using mongoDB drivers.

There are drivers available for php, node.js, C#, Java, Python, etc.

The drivers are used for CRUD operations.

Connect driver with mongodb database.

**Installing MongoDB on windows**:

We want to start mongo.exe and mongod.exe files using commands rather than manually going to the installation directories.

Don’t want to do cd into the directories.

The idea is to run mongod command regardless of the current directory in which we are located in.

Need to set environment variables for mongodb on our operating system.

Add C:\mongodb\bin in the Path.

If we type “mongo” and “mongod” on powershell, then operating system is able to find the .exe files

Git bash vs powershell

Git bash can emulate the bash terminal

Cmd vs Git bash vs powershell

A terminal application is sending our keystrokes to a shell generally and the shell is responsible for understanding our keystrokes and executing the commands that we are requesting.

In Windows there is cmd shell

On linux and mac, one of the popular shell is bash.

Powershell is a whole new approach to command line shell technology.

Git bash emulates a bash environment on Windows. It lets us use all git features in command line plus most of standard unix commands.