**What is an error-first callback?**

Error-first callbacks are used to pass errors and data as well. You have to pass the error as the first parameter, and it has to be checked to see if something went wrong. Additional arguments are used to pass data.

fs.readFile(filePath, **function**(err, data) {

**if** (err) {

*// handle the error, the return is important here*

*// so execution stops here*

**return** console.log(err)

}

*// use the data object*

console.log(data)

})

**How can you avoid callback hells?**

There are lots of ways to solve the issue of callback hells:

* **modularization**: break callbacks into independent functions
* use a **control flow library**, like [async](https://www.npmjs.com/package/async)
* use **generators with Promises**
* use **async/await** *(note that it is only available in the latest v7 release and not in the LTS version -*[*you can read our experimental async/await how-to here*](https://blog.risingstack.com/async-await-node-js-7-nightly/)*)*

[**Q: How to avoid callback hells? A: modularization, control flow libraries, generators with promises, async/await**](https://twitter.com/share?text=Q%3AHow%20to%20avoid%20callback%20hells%3F%20A%3A%20modularization%2C%20control%20flow%20libraries%2C%20generators%20with%20promises%2C%20async%2Fawait;url=https://blog.risingstack.com/node-js-interview-questions-and-answers-2017)

[CLICK TO TWEET](https://twitter.com/share?text=Q%3A%20How%20to%20avoid%20callback%20hells%3F%20A%3A%20modularization%2C%20control%20flow%20libraries%2C%20generators%20with%20promises%2C%20async%2Fawait;url=https://blog.risingstack.com/node-js-interview-questions-and-answers-2017)

**What are Promises?**

Promises are a concurrency primitive, first described in the 80s. Now they are part of most modern programming languages to make your life easier. Promises can help you better handle async operations.

An example can be the following snippet, which after 100ms prints out the resultstring to the standard output. Also, note the catch, which can be used for error handling. Promises are chainable.

**new** Promise((resolve, reject) => {

setTimeout(() => {

resolve('result')

}, 100)

})

.then(console.log)

.catch(console.error)

**What tools can be used to assure consistent style? Why is it important?**

When working in a team, consistent style is important, so team members can modify more projects easily, without having to get used to a new style each time.

Also, it can help eliminate programming issues using static analysis.

Tools that can help:

* [ESLint](http://eslint.org/)
* [Standard](http://standardjs.com/)

If you’d like to be even more confident, I suggest you to learn and embrace the [JavaScript Clean Coding](https://blog.risingstack.com/javascript-clean-coding-best-practices-node-js-at-scale/) principles as well!

**What's a stub? Name a use case!**

Stubs are functions/programs that simulate the behaviors of components/modules. Stubs provide canned answers to function calls made during test cases.

An example can be writing a file, without actually doing so.

**var** fs = require('fs')

**var** writeFileStub = sinon.stub(fs, 'writeFile', **function** (path, data, cb) {

**return** cb(null)

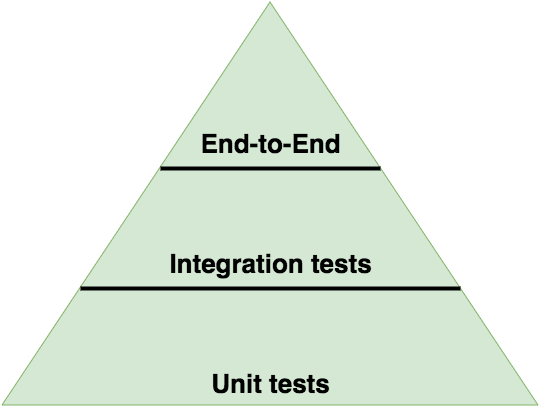
})

expect(writeFileStub).to.be.called

writeFileStub.restore()

**What's a test pyramid? Give an example!**

A test pyramid describes the ratio of how many unit tests, integration tests and end-to-end test you should write.



An example for an HTTP API may look like this:

* lots of low-level unit tests for models *(dependencies****are stubbed****)*,
* fewer integration tests, where you check how your models interact with each other *(dependencies****are not stubbed****)*,
* less end-to-end tests, where you call your actual endpoints *(dependencies****are not stubbed****)*.

**What's your favorite HTTP framework and why?**

There is no right answer for this. The goal here is to understand how deeply one knows the framework she/he uses. Tell what are the pros and cons of picking that framework.

**When are background/worker processes useful? How can you handle worker tasks?**

Worker processes are extremely useful if you'd like to do data processing in the background, like sending out emails or processing images.

There are lots of options for this like [RabbitMQ](https://www.rabbitmq.com/) or [Kafka](https://kafka.apache.org/).

**How can you secure your HTTP cookies against XSS attacks?**

XSS occurs when the attacker injects executable JavaScript code into the HTML response.

To mitigate these attacks, you have to set flags on the set-cookie HTTP header:

* **HttpOnly** - this attribute is used to help prevent attacks such as cross-site scripting since it does not allow the cookie to be accessed via JavaScript.
* **secure** - this attribute tells the browser to only send the cookie if the request is being sent over HTTPS.

So it would look something like this: Set-Cookie: sid=<cookie-value>; HttpOnly. If you are using Express, with [express-cookie session](https://github.com/expressjs/cookie-session#cookie-options), it is working by default.

**How can you make sure your dependencies are safe?**

When writing Node.js applications, **ending up with hundreds or even thousands of dependencies can easily happen**.  
For example, if you depend on Express, you depend on [27 other modules](https://github.com/expressjs/express/blob/master/package.json#L29) directly, and of course on those dependencies' as well, so manually checking all of them is not an option!

The only option is to automate the update / security audit of your dependencies. For that there are free and paid options:

* npm outdated
* [Trace by RisingStack](https://trace.risingstack.com/)
* [NSP](https://nodesecurity.io/)
* [GreenKeeper](https://greenkeeper.io/)
* [Snyk](https://snyk.io/)

**Node.js Interview Puzzles**

The following part of the article is useful if you’d like to prepare for an interview that involves puzzles, or tricky questions.

**What's wrong with the code snippet?**

**new** Promise((resolve, reject) => {

**throw** **new** Error('error')

}).then(console.log)

**The Solution**

As there is no catch after the then. This way the error will be a silent one, there will be no indication of an error thrown.

To fix it, you can do the following:

**new** Promise((resolve, reject) => {

**throw** **new** Error('error')

}).then(console.log).catch(console.error)

If you have to debug a huge codebase, and you don't know which Promise can potentially hide an issue, you can use the unhandledRejection hook. It will print out all unhandled Promise rejections.

process.on('unhandledRejection', (err) => {

console.log(err)

})

**What's wrong with the following code snippet?**

**function** **checkApiKey** (apiKeyFromDb, apiKeyReceived) {

**if** (apiKeyFromDb === apiKeyReceived) {

**return** true

}

**return** false

}

**The Solution**

When you compare security credentials it is crucial that you don't leak any information, so you have to make sure that you compare them in fixed time. If you fail to do so, your application will be vulnerable to [timing attacks](https://en.wikipedia.org/wiki/Timing_attack).

But why does it work like that?

**V8, the JavaScript engine used by Node.js, tries to optimize the code you run from a performance point of view.** It starts comparing the strings character by character, and once a mismatch is found, it stops the comparison operation. **So the longer the attacker has right from the password, the more time it takes.**

To solve this issue, you can use the npm module called [cryptiles](https://www.npmjs.com/package/cryptiles).

**function** **checkApiKey** (apiKeyFromDb, apiKeyReceived) {

**return** cryptiles.fixedTimeComparison(apiKeyFromDb, apiKeyReceived)

}

**What's the output of following code snippet?**

Promise.resolve(1)

.then((x) => x + 1)

.then((x) => { **throw** **new** Error('My Error') })

.catch(() => 1)

.then((x) => x + 1)

.then((x) => console.log(x))

.catch(console.error)

**The Answer**

The short answer is 2 - however with this question **I'd recommend asking the candidates to explain what will happen line-by-line to understand how they think**. It should be something like this:

1. A new Promise is created, that will resolve to 1.
2. The resolved value is incremented with 1 (so it is 2 now), and returned instantly.
3. The resolved value is discarded, and an error is thrown.
4. The error is discarded, and a new value (1) is returned.
5. The execution did not stop after the catch, but before the exception was handled, it continued, and a new, incremented value (2) is returned.
6. The value is printed to the standard output.
7. This line won't run, as there was no exception.

**Q1. What is Node.js? What is it used for?**

Node.js is a run-time JavaScript environment built on top of Chrome’s V8 engine. It uses an event-driven, non-blocking I/O model. It is lightweight and so efficient. Node.js has a package ecosystem called **npm**.

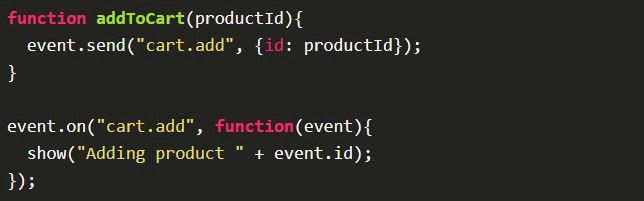
Node.js can be used to build different types of applications such as web application, real-time chat application, REST API server etc. However, it is mainly used to build network programs like web servers, similar to PHP, Java, or ASP.NET. Node.js was developed by Ryan Dahl in 2009.

**Q2. What is Event-driven programming?**

Event-driven programming is building our application based on and respond to events. When an event occurs, like click or keypress, we are running a callback function which is registered to the element for that event.

Event driven programming follows mainly a publish-subscribe pattern.

https://miro.medium.com/max/60/1*CIivQrkknaTktQTTMlK1Nw.jpeg?q=20



**Q3. What is *Event loop* in Node.js work? And How does it work?**

The *Event loop* handles all async callbacks. Node.js (or JavaScript) is a single-threaded, event-driven language. This means that we can attach listeners to events, and when a said event fires, the listener executes the callback we provided.

Whenever we are call setTimeout, http.get and fs.readFile, Node.js runs this operations and further continue to run other code without waiting for the output. When the operation is finished, it receives the output and runs our callback function.

So all the callback functions are queued in an loop, and will run one-by-one when the response has been received.

**Q4. What is REPL in Node.js?**

REPL means Read-Eval-Print-Loop. It is a virtual environment that comes with Node.js. We can quickly test our JavaScript code in the Node.js REPL environment.

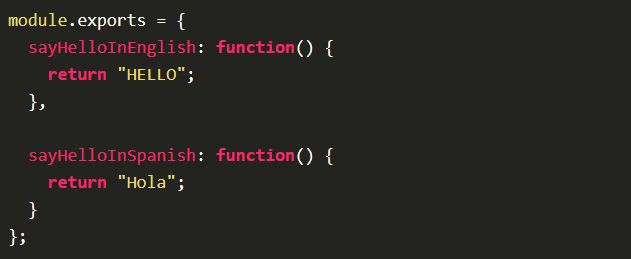
To launch the REPL in Node.js, just open the command prompt and type node. It will change the prompt to > in Windows and MAC.

Now we can type and run our JavaScript easily. For example, if we type 10 + 20, it will print 30 in the next line.

**Q5. What is the purpose of module.exports in Node.js?**

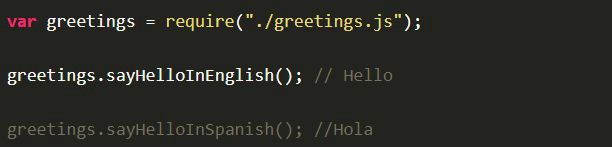
A module encapsulates related code into a single unit of code. This can be interpreted as moving all related functions into a file. Imagine that we created a file called greetings.js and it contains the following two functions:

https://miro.medium.com/max/60/1*oI0dKGDLPA0yGbg6GIWKsQ.jpeg?q=20



In the above code, module.exports exposes two functions to the outer world. We can import them in another file as follow:

https://miro.medium.com/max/60/1*2UOu3-O4GW3sN-NiJGhz2Q.jpeg?q=20



**Q6. What is the difference between Asynchronous and Non-blocking?**

Asynchronous literally means not synchronous. We are making HTTP requests which are asynchronous, means we are not waiting for the server response. We continue with other block and respond to the server response when we received.

The term Non-Blocking is widely used with IO. For example non-blocking read/write calls return with whatever they can do and expect caller to execute the call again. Read will wait until it has some data and put calling thread to sleep.

**Q7. What is Tracing in Node.js?**

Tracing provides a mechanism to collect tracing information generated by V8, Node core and userspace code in a log file. Tracing can be enabled by passing the --trace-events-enabled flag when starting a Node.js application.

https://miro.medium.com/max/60/1*gaM6bjJjwacw6o861eTKBA.jpeg?q=20

https://miro.medium.com/max/700/1*gaM6bjJjwacw6o861eTKBA.jpeg

The set of categories for which traces are recorded can be specified using the --trace-event-categories flag followed by a list of comma separated category names. By default the node and v8 categories are enabled.

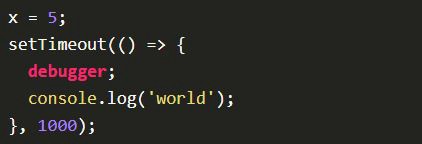
Running Node.js with tracing enabled will produce log files that can be opened in the chrome://tracing tab of Chrome.

**Q8. How will you debug an application in Node.js?**

Node.js includes a debugging utility called debugger. To enable it start the Node.js with the debug argument followed by the path to the script to debug.

Inserting the statement debugger; into the source code of a script will enable a breakpoint at that position in the code:

https://miro.medium.com/max/60/1*qy5HcWVLICvVaRLI_-pt9w.jpeg?q=20



**Q9. Difference between setImmediate() vs setTimeout()**

setImmediate() and setTimeout() are similar, but behave in different ways depending on when they are called.

* setImmediate() is designed to execute a script once the current poll (event loop) phase completes.
* setTimeout() schedules a script to be run after a minimum threshold in ms has elapsed.

The order in which the timers are executed will vary depending on the context in which they are called. If both are called from within the main module, then timing will be bound by the performance of the process.

**Q10. What is process.nextTick()**

setImmediate() and setTimeout() are based on the event loop. But process.nextTick() technically not part of the event loop. Instead, the nextTickQueue will be processed after the current operation completes, regardless of the current phase of the event loop.

Thus, any time you call process.nextTick() in a given phase, all callbacks passed to process.nextTick() will be resolved before the event loop continues.

**Q11. What is package.json? What is it used for?**

This file holds various metadata information about the project. This file is used to give information to npm that allows it to identify the project as well as handle the project's dependencies.

Some of the fields are: name, name, description, author and dependencies.

When someone installs our project through npm, all the dependencies listed will be installed as well. Additionally, if someone runs npm install in the root directory of our project, it will install all the dependencies to ./node\_modules directory.

**Q12. What is libuv?**

libuv is a multi-platform support library with a focus on asynchronous I/O.It was primarily developed for use by Node.js, but it’s also used by Luvit, Julia, pyuv, and others.

When the node.js project began in 2009 as a JavaScript environment decoupled from the browser, it is using Google’s V8 and Marc Lehmann’s libev, node.js combined a model of I/O – evented – with a language that was well suited to the style of programming; due to the way it had been shaped by browsers. As node.js grew in popularity, it was important to make it work on Windows, but libev ran only on Unix. libuv was an abstraction around libev or IOCP depending on the platform, providing users an API based on libev. In the node-v0.9.0 version of libuv libev was removed.

Some of the features of libuv are:

* Full-featured event loop backed by epoll, kqueue, IOCP, event ports.
* Asynchronous TCP and UDP sockets
* Asynchronous file and file system operations
* Child processes
* File system events

**Q13. What are some of the most popular modules of Node.js?**

There are many most popular, most starred or most downloaded modules in Node.js. Some of them are:

* express
* async
* browserify
* socket.io
* bower
* gulp
* grunt

**Q14. What is EventEmitter in Node.js?**

All objects that emit events are instances of the EventEmitter class. These objects expose an eventEmitter.on() function that allows one or more functions to be attached to named events emitted by the object.

When the EventEmitter object emits an event, all of the functions attached to that specific event are called *synchronously*.

https://miro.medium.com/max/60/1*Te3ctZkSEqXKFGhj1Tn9MA.jpeg?q=20



**Q15. What is Streams in Node.js?**

Streams are pipes that let you easily read data from a source and pipe it to a destination. Simply put, a stream is nothing but an EventEmitter and implements some specials methods. Depending on the methods implemented, a stream becomes Readable, Writable, or Duplex (both readable and writable).

For example, if we want to read data from a file, the best way to do it from a stream is to listen to data event and attach a callback. When a chunk of data is available, the readable stream emits a data event and your callback executes. Take a look at the following snippet:

https://miro.medium.com/max/60/1*CPRmtq0khNvHLcJX7F09iA.jpeg?q=20



Types of streams are: Readable, Writable, Duplex and Transform.

**Q16**. **What is the difference between readFile vs createReadStream in Node.js?**

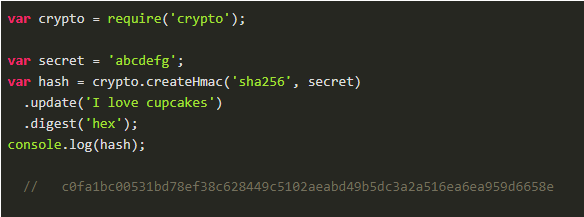
**readFile** — is for asynchronously reads the entire contents of a file. It will read the file completely into memory before making it available to the User. readFileSync is synchronous version of readFile.

**createReadStream** — It will read the file in chunks of the default size 64 kb which is specified before hand.

**Q17. What is crypto in Node.js? How do you cipher the secured information in Node.js?**

The crypto module in Node.js provides cryptographic functionality that includes a set of wrappers for OpenSSL's hash, HMAC, cipher, decipher, sign and verify functions.

https://miro.medium.com/max/60/1*usB3iZ9uwyP3uMnyb8GY4w.png?q=20



**Q18. What is the use of Timers is Node.js?**

The Timers module in Node.js contains functions that execute code after a set period of time. Timers do not need to be imported via require(), since all the methods are available globally to emulate the browser JavaScript API.

The Node.js API provides several ways of scheduling code to execute at some point after the present moment. The functions below may seem familiar, since they are available in most browsers, but Node.js actually provides its own implementation of these methods.

Node.js Timer provides setTimeout(), setImmediate() and setInterval.

**Q19. What is the use of DNS module in Node.js?**

dns module which provide underlying system's name resolution and DNS look up facilities. DNS module consists of an asynchronous network wrapper.

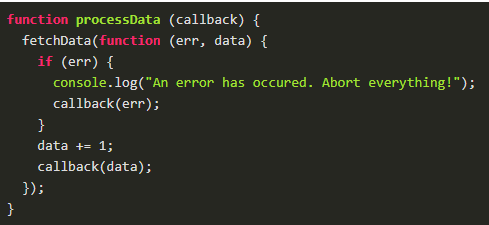
The most commonly used functions in DNS module are:

* **dns.lookup(adress, options, callback)** - The dns lookup method takes any website address as its first parameter and returns the corresponding first IPV4 or IPV6 record. The options parameter can be an integer or object. If no options are provided both IPV4 and IPV6 are valid inputs. The third parameter is the callback functions.
* **dns.lookupservice(address, port, callback)** - This function converts any physical address such as “www.knowledgehills.com” to array of record types. The record types are specified by the second parameter “rrbyte”. Finally the third method is the callback function.
* **dns.getServers()** - This function returns an array of IP address strings, formatted according to rfc5952, that are currently configured for DNS resolution. A string will include a port section if a custom port is used.
* **dns.setServers()** - This function sets the IP address and port of servers to be used when performing DNS resolution. The dns.setServers() method must not be called while a DNS query is in progress.

**Q20. What is a Callback function in Node.js?**

Node.js, being an asynchronous platform, doesn’t wait around for things like file I/O to finish — Node.js uses callbacks. A callback is a function called at the completion of a given task; this prevents any blocking, and allows other code to be run in the meantime.

https://miro.medium.com/max/60/1*-cFWHUCI634Aln30S4MaRA.png?q=20



Callbacks are the foundation of Node.js. Callbacks give us an interface with which to say, “and when you’re done doing that, do all this.” This allows us to have as many IO operations as our OS can handle happening at the same time. For example, in a web server with hundreds or thousands of pending requests with multiple blocking queries, performing the blocking queries asynchronously gives you the ability to be able to continue working and not just sit still and wait until the blocking operations come back.

**Q21. What are the security mechanisms available in Node.js?**

We can secure our Node.js application in the following ways:

**Authentication** — Authentication is one of the primary security stages at which user is identified as permitted to access the application at all. Authentication verifies the user’s identity through one or several checks. In Node.js, authentication can be either session-based or token-based. In session-based authentication, the user’s credentials are compared to the user account stored on the server and, in the event of successful validation, a session is started for the user. Whenever the session expires, the user needs to log in again. In token-based authentication, the user’s credentials are applied to generate a string called a token which is then associated with the user’s requests to the server.

**Error Handling** — Usually, the error message contains the explanation of what’s actually gone wrong for the user to understand the reason. At the same time, when the error is related to the application code syntax, it can be set to display the entire log content on the frontend. For an experienced hacker, the log content can reveal a lot of sensitive internal information about the application code structure and tools used within the software.

**Request Validation** — Another aspect which has to be considered, while building a secure Node.js application, is a validation of requests or, in other words, a check of the incoming data for possible inconsistencies. It may seem that invalid requests do not directly affect the security of a Node.js application, however, they may influence its performance and robustness. Validating the incoming data types and formats and rejecting requests not conforming to the set rules can be an additional measure of securing your Node.js application.

**Node.js Security Tools and Best Practices** — We can use tools like **helmet**(protects our application by setting HTTP headers), **csurf** (validates tokens in incoming requests and rejects the invalid ones), **node rate limiter**(controls the rate of repeated requests. This function can protect you from brute force attacks) and **cors** (enables cross-origin resource sharing).

**Q22. What is the passport in Node.js?**

Passport.js is a simple, unobtrusive Node.js authentication middleware for Node.js. Passport.js can be dropped into any Express.js-based web application.

Passport recognizes that each application has unique authentication requirements. Authentication mechanisms, known as strategies, are packaged as individual modules. Applications can choose which strategies to employ, without creating unnecessary dependencies.

By default, if authentication fails, Passport will respond with a 401 Unauthorized status, and any additional route handlers will not be invoked. If authentication succeeds, the next handler will be invoked and the req.user property will be set to the authenticated user.

1. What is Node.js? Where can you use it?

Node.js is a server side scripting based on Google’s V8 JavaScript engine. It is used to build scalable programs especially web applications that are computationally simple but are frequently accessed.

You can use [Node.js in developing](https://www.simplilearn.com/developing-java-and-node-js-application-with-mongodb-tutorial-video) I/O intensive web applications like video streaming sites. You can also use it for developing: Real-time web applications, Network applications, General-purpose applications and Distributed systems.

2. Why use Node.js?

Node.js makes building scalable network programs easy. Some of its advantages include:

* It is generally fast
* It almost never blocks
* It offers a unified programming language and data type
* Everything is asynchronous
* It yields great concurrency

3. What are the features of Node.js?

Node.js is a single-threaded but highly scalable system that utilizes JavaScript as its scripting language. It uses asynchronous, event-driven I/O instead of separate processes or threads. It is able to achieve high output via single-threaded event loop and non-blocking I/O.

4. How else can the JavaScript code below be written using Node.Js to produce the same output?

console.log("first");  
setTimeout(function() {  
    console.log("second");  
}, 0);  
console.log("third");

Output:

first  
third  
second

In Node.js version 0.10 or higher, setImmediate(fn) will be used in place of setTimeout(fn,0) since it is faster. As such, the code can be written as follows:

console.log("first");  
setImmediate(function(){  
    console.log("second");  
});  
console.log("third");

5. How do you update NPM to a new version in Node.js?

You use the following commands to update NPM to a new version:

$ sudo npm install npm -g  
/usr/bin/npm -> /usr/lib/node\_modules/npm/bin/npm-cli.js  
npm@2.7.1 /usr/lib/node\_modules/npm

6. Why is Node.js Single-threaded?

Node.js is single-threaded for async processing. By doing async processing on a single-thread under typical web loads, more performance and scalability can be achieved as opposed to the typical thread-based implementation.

7. Explain callback in Node.js.

A callback function is called at the completion of a given task. This allows other code to be run in the meantime and prevents any blocking.  Being an asynchronous platform, Node.js heavily relies on callback. All APIs of Node are written to support callbacks.

8. What is callback hell in Node.js?

Callback hell is the result of heavily nested callbacks that make the code not only unreadable but also difficult to maintain. For example:

query("SELECT clientId FROM clients WHERE clientName='picanteverde';", function(id){  
  query("SELECT \* FROM transactions WHERE clientId=" + id, function(transactions){  
    transactions.each(function(transac){  
      query("UPDATE transactions SET value = " + (transac.value\*0.1) + " WHERE id=" + transac.id, function(error){  
        if(!error){  
          console.log("success!!");  
        }else{  
          console.log("error");  
        }  
      });  
    });  
  });  
});

9. How do you prevent/fix callback hell?

The three ways to prevent/fix callback hell are:

* Handle every single error
* Keep your code shallow
* Modularize – split the callbacks into smaller, independent functions that can be called with some parameters then joining them to achieve desired results.

The first level of improving the code above might be:

var logError = function(error){  
    if(!error){  
      console.log("success!!");  
    }else{  
      console.log("error");  
    }  
  },  
  updateTransaction = function(t){  
    query("UPDATE transactions SET value = " + (t.value\*0.1) + " WHERE id=" + t.id, logError);  
  },  
  handleTransactions = function(transactions){  
    transactions.each(updateTransaction);  
  },  
  handleClient = function(id){  
    query("SELECT \* FROM transactions WHERE clientId=" + id, handleTransactions);  
  };

query("SELECT clientId FROM clients WHERE clientName='picanteverde';",handleClient);

You can also use Promises, Generators and Async functions to fix callback hell.

10. Explain the role of REPL in Node.js.

As the name suggests, REPL (Read Eval print Loop) performs the tasks of – Read, Evaluate, Print and Loop. The REPL in Node.js is used to execute ad-hoc Javascript statements. The REPL shell allows entry to javascript directly into a shell prompt and evaluates the results. For the purpose of testing, debugging, or experimenting, REPL is very critical.

11. Name the types of API functions in Node.js.

There are two types of functions in Node.js.:

* Blocking functions - In a blocking operation, all other code is blocked from executing until an I/O event that is being waited on occurs. Blocking functions execute synchronously

For example:  
const fs = require('fs');  
const data = fs.readFileSync('/file.md'); // blocks here until file is read  
console.log(data);  
// moreWork(); will run after console.log

The second line of code blocks the execution of additional JavaScript until the entire file is read. moreWork () will only be called after Console.log

* Non-blocking functions - In a non-blocking operation, multiple I/O calls can be performed without the execution of the program being halted.  Non-blocking functions execute asynchronously.

For example:

const fs = require('fs');  
fs.readFile('/file.md', (err, data) => {  
  if (err) throw err;  
  console.log(data);  
});  
// moreWork(); will run before console.log

Since fs.readFile () is non-blocking, moreWork () does not have to wait for the file read to complete before being called. This allows for higher throughput.

12. Which is the first argument typically passed to a Node.js callback handler?

Typically, the first argument to any callback handler is an optional error object. The argument is null or undefined if there is no error.

Error handling by a typical callback handler could be as follows:

function callback(err, results) {  
    // usually we'll check for the error before handling results  
    if(err) {  
        // handle error somehow and return  
    }  
    // no error, perform standard callback handling  
}

13. What are the functionalities of NPM in Node.js?

NPM (Node package Manager) provides two functionalities:

* Online repository for Node.js packages
* Command line utility for installing packages, version management and dependency management of Node.js packages

14. What is the difference between Node.js and Ajax?

Node.js and Ajax (Asynchronous JavaScript and XML) are the advanced implementation of JavaScript. They all serve completely different purposes.

Ajax is primarily designed for dynamically updating a particular section of a page’s content, without having to update the entire page.

Node.js is used for developing client-server applications.

15. Explain chaining in Node.js.

Chaining is a mechanism whereby the output of one stream is connected to another stream creating a chain of multiple stream operations.

16. What are “streams” in Node.js? Explain the different types of streams present in Node.js.

Streams are objects that allow reading of data from the source and writing of data to the destination as a continuous process.

There are four types of streams.

* <Readable> to facilitate the reading operation
* <Writable> to facilitate the writing operation
* <Duplex> to facilitate both read and write operations
* <Transform> is a form of Duplex stream that performs computations based on the available input

17. What are exit codes in Node.js? List some exit codes.

Exit codes are specific codes that are used to end a “process” (a global object used to represent a node process).

Examples of exit codes include:

* Unused
* Uncaught Fatal Exception
* Fatal Error
* Non-function Internal Exception Handler
* Internal Exception handler Run-Time Failure
* Internal JavaScript Evaluation Failure

18. What are Globals in Node.js?

Three keywords in Node.js constitute as Globals. These are:

* Global – it represents the Global namespace object and acts as a container for all other <global> objects.
* Process – It is one of the global objects but can turn a synchronous function into an async callback. It can be accessed from anywhere in the code and it primarily gives back information about the application or the environment.
* Buffer – it is a class in Node.js to handle binary data.

19. What is the difference between AngularJS and Node.js?

Angular.JS is a web application development framework while Node.js is a runtime system.

20. Why is consistent style important and what tools can be used to assure it?

Consistent style helps team members modify projects easily without having to get used to a new style every time. Tools that can help include Standard and ESLint.

**1. What is Node js ?**

**Node Js** is one of the most popular and powerful server technologies today.  
It allows you built the entire website only in one programming Language i.e Javascript. Node js is free and open source server technology that uses Javascript to create complete web software.It runs on various platforms like Windows, Linux, Unix, Mac OS X, etc.

**2. Explain Modules in Node Js ?**

**Modules** are reusable block of code whose existence does not impact other code in any way. It is not supported by Javascript. Modules are introduced in ES6. Modules are important for Maintainability, Reusability, and Namespacing of Code.

**3. Is Node Js Single-threaded ?**

Yes, Node Js is single threaded to perform asynchronous processing. Doing async processing on a single thread could provide more performance and scalability under typical web loads than the typical thread-based implementation.

**4. For what require() is used in Node Js ?**

**require()** is used to include modules from external files in Node Js. It is the easiest way to include a module in Node. Basically require is a function that takes a string parameter which contains the location of the file that you want to include. It reads the entire javascript file, executes the file, and then proceeds to return the exports object.  
Syntax:

require('path');

**5. In which Language Node Js is written ?**

**Node js** is written in C, C++,JavaScript.It uses Google’s open source V8 Javascript Engine to convert Javascript code to C++.

**6. List the types of application you can build using Node Js ?**

**Using Node Js you can build applications like:**

* Internet of Things
* Real-Time Chats Applications
* Complex Single-Page Applications
* Real-Time Collaboration Tools
* Streaming apps
* Microservices / API’s

**7. Explain what is libuv in Node Js ?**

libuv is Cross-platform I/O abstraction library that supports asynchronous I/O based on event loops.It is written in C and released under MIT Licence.

libuv support Windows IOCP, epoll(4), kqueue(2), and Solaris event ports. Initially, it was designed for Node.js but later it is also used by other software projects.

Reference : <https://en.wikipedia.org/wiki/Libuv>

**8. Why Zlib is used in Node js ?**

Zlib is Cross-platform data compression library. It was written by Jean-loup Gailly and Mark Adler. In Node js, you can Zlib for Threadpool, HTTP requests, and responses compression and Memory Usage Tuning. In order to use zlib in node js, you need to install node-zlib package. After installation below is sample code to use Zlib.

var Buffer = require('buffer').Buffer;

var zlib = require('zlib');

var input = new Buffer('lorem ipsum dolor sit amet');

var compressed = zlib.deflate(input);

var output = zlib.inflate(compressed);

Further Reading <https://nodejs.org/api/zlib.html>

<https://en.wikipedia.org/wiki/Zlib>

**9. How Node js read the content of a file?**

Normally NodeJs reads the content of a file in non-blocking, asynchronous way. Node Js uses its **fs** core API to deal with files. The easiest way to read the entire content of a file in nodeJs is with fs.readFile method. Below is sample code to read a file in NodeJs asynchronously and synchronously.

**Reading a file in node asynchronously/ non-blocking**

var fs = require('fs');

fs.readFile('DATA', 'utf8', function(err, contents) {

console.log(contents);

});

console.log('after calling readFile');

**Reading a file in node asynchronously/blocking**

var fs = require('fs');

var contents = fs.readFileSync('DATA', 'utf8');

console.log(contents);

**10. What are Streams? List types of streams available in Node Js ?**

Streams are special types of objects in Node that allow us to read data from a source or write data to a destination continuously. There are 4 types of streams available in Node Js, they are

* Readable − For reading operation.
* Writable − For writing operation.
* Duplex − Used for both read and write operation.
* Transform − A type of duplex stream where the output is computed based on the input.

Further reading: <https://www.tutorialspoint.com/nodejs/nodejs_streams.htm>

**11. What's the first argument passed to a Node Js callback handler?**

In Node Js all core modules, as well as most of the community-published modules, follows a pattern where the first argument to any callback handler is an error object. this object is optional, if there is no error then in that case null or undefined is passed to the callback.

**Example of the callback function**

function callback(err, results) {

// usually we'll check for the error before handling results

if(err) {

// handle error somehow and return

}

// no error, perform standard callback handling

}

**12. Explain the difference between readFile and createReadStream in Node js ?**

* readFile load the whole file which you had marked to read whereas createReadStream reads the complete file in the parts of the size you have declared.
* The client will receive the data faster in the case of createReadStream in contrast with readFile.
* In readFile, a file will first completely read by memory and then transfers to a client but in later option, a file will be read by memory in a part which is sent to clients and the process continue until all the parts finish.

**13. What is JIT and how is it related to Node JS ?**

JIT stands for Just-in-time. A JIT compiler is a program which is used to send bytecode (it consists of instruction that can be interpreted) to the processor by converting it into instruction. After you have done with writing a program, the compiler compiles the source language statements into bytecode instead of compiling it into the code that carries the information which is similar to the specific hardware platform's processor.

**Relation of JIT with Node**: Virtual machine of Nodejs has JIT compilation which improves the execution speed of the code. The virtual machine takes the source code and converts to machine code in runtime. By this, the hot functions which are called very often are compiled to machine code and, hence increasing speed.

**14. How to create a simple server in Node js that returns Hello World ?**

By writing following line of code, you can **create a server in Node Js**.

var http =require('http');

http.createServer(function(req,res){

res.writeHead(200,{'Content-Type':'text/plain'});

res.end('Hello World\n');

}).listen(1320,'127.0.0.3');

**15. what is Closures?**

A Closure is a function defined within another scope that has access to all the variables within the outer scope.

Global variables can be made local (private) with closures.

**16. How to use aggregation in Mongoose?**

Aggregations are a set of functions that are used to manipulate the data that has been returned from a MongoDB query. In Mongoose, aggregations work as a pipeline. The aggregate function accepts the array of data transformations which are applied by data using different methods in terms of arguments.

Syntax: db.customers.aggregate([ ... aggregation steps go here ...]);

In above, aggregation is applied to data of customers.

**17. What is difference between put and patch?**

The main difference between the put and the patch is

|  |  |
| --- | --- |
| **Put** | **Patch** |
| The embedded entity is believed to the modified version of the resources that are deposited on the original server. It is requested to the client to replace the stored is substituted. | In this, the information regarding the way of modifying the original server which has the resources to produce a new version is found. |
| At the time of updating the resource, you need to forward full payload as the request. | At the time of updating the resource, you only need to send the parameter of the resource which you want to update. |

**18. List types of Http requests?**

Http defines a set of request methods to perform the desired actions. These request methods are:

1. **GET**: The GET method asked for the representation of the specifies resource. This request used to retrieve the data.
2. **POST**: The POST technique is utilized to present an element to the predetermined resource, generally causing a change in state or reactions on the server.
3. **HEAD**: The HEAD method is similar to the GET method but asks for the response without the response body.
4. **PUT**: This method is used to substitute all current representations with the payload.
5. **DELETE**: It is used to delete the predetermined resource.
6. **CONNECT**: This request is used to settle the TCP/IP tunnel to the server by the target resource
7. **OPTION**: This method is used to give back the HTTP strategies to communicate with the target resource.
8. **TRACE**: This method echoes the message which tells the customer how much progressions have been made by an intermediate server.
9. **PATCH**: The PATCH method gives partial modifications to a resource.

**19. What is the revealing module pattern?**

Revealing module pattern is similar to Module Pattern.IT ExposES only the properties and methods you want via an returned Object.

var greeting = 'Hello world'

function greet() {

console.log(greeting)

}

module.exports = {

greet: greet

}

**20. What are CommonJs Modules ?**

CommonJS Modules is the Standard how to code modules are structured. It specifies an ecosystem for JavaScript outside on the server or for native desktop applications.

**21. Explain the difference between process.tick() and setImmediate() ?**

**22. How can models be defined in Express JS?**

There is no notion of any database in Express JS. So, the concept of models is left up to third-party node modules, allowing the users to interface with nearly any type of database.

**23. How to config properties in Express JS?**

In Express JS, there are two ways for configuring the properties:

* With process.ENV:
* A file with the name “.env” is to be created inside the project folder.
* All the properties are to be added in the “.env” file.
* Any of the properties can be used in server.js.
* With require JS:
* A file with the name “config.json” is to be created in the config folder inside the project folder.
* The config properties are to be added in the config.json file.
* Now, require should be used to access the config.json file.

**24. How to authenticate users in express JS?**

Since authentication is an opinionated area which is not ventured by express JS, therefore any authentication scheme can be used in express JS for the authentication of users.

**25. What function are arguments available to Express JS route handlers?**

The arguments which are available to an Express JS route handler-function are-

* Req – the request object
* Res – the response object
* Next (optional) – a function which is used to pass control to one of the subsequent route handlers.

The third argument is optional and may be omitted, but in some cases, it is useful where there is a chain of handlers and control can be passed to one of the subsequent route handlers skipping the current one.

**26. Which template engine is supported by express JS?**

Express JS supports any template engine that conforms to the (path, locals, callback) signature.

**27. Name the type of web applications which can be built using Express JS.**

Single-page, multi-page, and hybrid web applications can be built using Express JS.

**28. How can plain HTML be rendered in express JS?**

There’s no need to render HTML with the res.render () function. If there’s a specific file, then you should use the res.sendFile () function. If any assets are being served from a dictionary, then express.static () middleware function needs to be used.

**29. What is the use of Express JS?**

Express.js is a lightweight web application which helps in organizing the web application into MVC architecture on the server side.

**30. Why to use Express.js?**

Below are the few reasons why to use Express with Node.js

* Express js is built on top of Node.js. It is the perfect framework for ultra-fast Input / Output.
* Cross Platform
* Support MVC Design pattern
* Support of NoSQL databases out of the box.
* Multiple templating engine support i.e. Jade or EJS which reduces the amount of HTML code you have to write for a page.
* Support Middleware, basic web-server creation, and easy routing tools.

**31. Write the steps for setting up an Express JS application.**

Following are the steps used to set up an express JS application: –

1. A folder with the same name as the project name is created.
2. A file named package.json is created inside the folder created.
3. “npm install” command is run on the command prompt. It installs all the libraries present in package.json.
4. A file named server.js is created.
5. “Router” file is created inside the package which consists of a folder named index.js.
6. “App” is created inside the package which has the index.html file.

This way, an express JS application is set up.

**32. List some features of Express JS.**

Some of the main features of Express JS are listed below: –

* It is used for setting up middlewares so as to provide a response to the HTTP or RESTful requests.
* With the help of express JS, the routing table can be defined for performing various HTTP operations.
* It is also used for dynamically rendering [HTML](https://www.onlineinterviewquestions.com/html-interview-questions/)pages which are based on passing arguments to the templates.
* It provides each and every feature which is provided by core Node JS.
* The performance of Express JS is adequate due to the presence of a thin layer prepared by the Express JS.
* It is used for organizing the web applications into the MVC architecture.
* Everything from routes to rendering view and performing HTTP requests can be managed by Express JS.

**33. What do you understand by middleware? How can you use middleware in Node Js?**

**34. How does Promise and Queue work?**

**35. What is difference between return and callback in JavaScript functions?**

**36. Describe Node.js event loop and event driver architecture?**

**37. How Promises are better than callbacks?**

**38. List out some new features introduced in ES6?**

Following are the list of few new Features introduced in ES6

* const and let keywords
* Array helper functions like map, forEach, filter, find, every, some, reduce
* Arrow functions
* Classes and enhanced object literals
* Template strings
* Default function arguments
* Rest and spread operators
* Promises
* Modules
* Multi-line Strings
* Destructuring Assignment

**39. Write a program to Print 0 to N element in pyramid shape?**

**40. Write a simple code to enable CORS in Node js?**

**CORS** stands for Cross-Origin Resource Sharing. It a is a mechanism that uses additional HTTP headers to tell a browser to let a web application running at one origin (domain) have permission to access selected resources from a server at a different origin.

**Use below code to enable CORS on NodeJS**

app.use(function(req, res, next) {

res.header("Access-Control-Allow-Origin", "\*");

res.header("Access-Control-Allow-Headers", "Origin, X-Requested-With, Content-Type, Accept");

next();

});

**41. How to generate unique UUIDs/ guid in Node Js**

Use node-uuid package to generate unique UUIDs/ guid in Node Js. Below code demonstrates how to generate it.

var uuid = require('node-uuid');

// Generate a v1 (time-based) id

uuid.v1();

// Generate a v4 (random) id

uuid.v4();

**42. What do you mean by Express JS?**

Express JS is an application framework which is light-weighted node JS. A number of flexible, useful and important features are provided by this JavaScript framework for the development of mobile as well as web applications with the help of node JS.

**43. How can you set default node version using nvm?**

Run below command on the terminal to set default node version along multiple installed versions of node. You can list all install versions of the node by running nvm ls

nvm alias default v7.3.0

**44. What are events ?**

An event is an action or occurrence recognized by software/app that is handled by event handler by writing a code that will be executed when the event fired.  
Mouse move, Click, file copied or deleted are some examples of events.  
In Node Js there are two types of events.  
1)System Events: The event that comes from the C++ side.  
2)Custom Events: Custom events are user-defined events.

**45. Explain event loop in Node Js ?**

In Node Js processes are single threaded, to supports concurrency it uses events and callbacks. An event loop is a mechanism that allows Node.js to perform non-blocking I/O operations.

**46. Difference between cluster and child\_process modules?**

**47. How to stop master process without suspending all of its child processes?**

**48. What does emitter do and what is dispatcher?**

**49. Since node is a single threaded process, how to make use of all CPUs?**

**50. Explain CLI.**

CLI stands for Command Line Interface. It is a utility or program on your computer where users type commands to perform some action or run some script rather than clicking on the screen.  
There are different types of command line interfaces depending on which operating system you are using. We have listed some of them below.

* Bash on Linux.
* Terminal of Mac.
* Command Prompt or Powershell on Windows
* Shell/ Command line/terminal on Unix and Ubuntu

**51. Who is the author of Node Js ?**

Node Js is written by Ryan Dahl.

**52. Explain What is a Javascript Engine ?**

A Javascript Engine is a program that converts code written in Javascript to something that computer processor understands.

**53. Explain V8 Engine ?**

V8 is Google’s open source high-performance JavaScript engine, written in C++ and used in Google Chrome, the open source browser from Google, and in Node.js, among [others](https://www.onlineinterviewquestions.com/others/). It implements ECMAScript as specified in ECMA-262, and runs on Windows 7 or later, macOS 10.5+, and Linux systems that use IA-32, ARM, or MIPS processors. V8 can run standalone or can be embedded into any C++ application.  
Source: https://developers.google.com/v8/

**54. Explain ECMAScript ?**

ECMAScript is the standard on which Javascript is based on. It was created to standardize Javascript. It is commonly used for client-side scripting on the World Wide Web and used by Node Js for writing server applications and services.

**55. How can you check the installed version of Node Js ?**

Use node -v command to check the installed version of Node Js.

**56. Explain What is NPM ?**

NPM stands for node package manager. It is default Package Manager for JavaScript programming language. NPM is used for installing/updating packages and modules of Javascript.

**57. Explain module.exports in Node Js ?**

The method or variable defined in modules cannot be directly accessible by the outer world, that means you cannot call a module member from the external file. In order to access module member, we need to export the functions or variables of modules using module.exports method.

Syntax and usage:

// greet.js

var greet=function(){

console.log("hello World");

}

module.exports=greet;

//In app.js

var greet=require('./greet.js');

greet();

Express.js is a free and open source software and web application framework for Node JS which was designed for the development of web applications and API’s. It is also the standard server framework for Node JS. It was developed by TJ Holowaychuk on 16th November 2010. Along with the [MongoDB](https://www.onlineinterviewquestions.com/mongodb-interview-questions/)database and [Angular JS frontend framework](https://www.onlineinterviewquestions.com/angularjs/), express JS is the backend part of the MEAN stack. With the help of this, you will be able to manage everything from routes to handling requests and views. It even provides a robust set of features for various web and mobile applications.

Q-1. What Is Node.Js?

**Answer.**

Node.js is a JavaScript runtime or platform which is built on Google Chrome’s JavaScript v8 engine. This runtime allows executing the JavaScript code on any machine outside a browser (this means that it is the server that executes the Javascript and not the browser).

Node.js is single-threaded, that employs a concurrency model based on an event loop. It doesn’t block the execution instead registers a callback which allows the application to continue. It means Node.js can handle concurrent operations without creating multiple threads of execution so can scale pretty well.

It uses JavaScript along with C/C++ for things like interacting with the filesystem, starting up HTTP or TCP servers and so on. Due to it’s extensively fast growing community and NPM, Node.js has become a very popular, open source and cross-platform app. It allows developing very fast and scalable network app that can run on Microsoft Windows, Linux, or OS X.

Following are the areas where it’s perfect to use Node.js.

* I/O bound Applications
* Data Streaming Applications
* Data Intensive Real-time Applications (DIRT)
* JSON APIs based Applications
* Single Page Applications

At the same time, it’s not suitable for heavy applications involving more of CPU usage.

Q-2. What Are The Key Features Of Node.Js?

**Answer.**

Let’s look at some of the key features of Node.js.

* **Asynchronous event driven IO helps concurrent request handling –** All APIs of Node.js are asynchronous. This feature means that if a Node receives a request for some Input/Output operation, it will execute that operation in the background and continue with the processing of other requests. Thus it will not wait for the response from the previous requests.
* **Fast in Code execution –** Node.js uses the V8 JavaScript Runtime engine, the one which is used by Google Chrome. Node has a wrapper over the JavaScript engine which makes the runtime engine much faster and hence processing of requests within Node.js also become faster.
* **Single Threaded but Highly Scalable –** Node.js uses a single thread model for event looping. The response from these events may or may not reach the server immediately. However, this does not block other operations. Thus making Node.js highly scalable. Traditional servers create limited threads to handle requests while Node.js creates a single thread that provides service to much larger numbers of such requests.
* **Node.js library uses JavaScript –** This is another important aspect of Node.js from the developer’s point of view. The majority of developers are already well-versed in JavaScript. Hence, development in Node.js becomes easier for a developer who knows JavaScript.
* **There is an Active and vibrant community for the Node.js framework –** The active community always keeps the framework updated with the latest trends in the web development.
* **No Buffering –** Node.js applications never buffer any data. They simply output the data in chunks.

Q-3. Explain How Do We Decide, When To Use Node.Js And When Not To Use It?

**Answer.**

When Should We Use Node.Js?

It’s ideal to use Node.js for developing streaming or event-based real-time applications that require less CPU usage such as.

* Chat applications.
* Game servers.

Node.js is good for fast and high-performance servers, that face the need to handle thousands of user requests simultaneously.

Good For A Collaborative Environment.

It is suitable for environments where multiple people work together. For example, they post their documents, modify them by doing check-out and check-in of these documents.

Node.js supports such situations by creating an event loop for every change made to the document. The “Event loop” feature of Node.js enables it to handle multiple events simultaneously without getting blocked.

Advertisement Servers.

Here again, we have servers that handle thousands of request for downloading advertisements from a central host. And Node.js is an ideal solution to handle such tasks.

Streaming Servers.

Another ideal scenario to use Node.js is for multimedia streaming servers where clients fire request’s towards the server to download different multimedia contents from it.

To summarize, it’s good to use Node.js, when you need high levels of concurrency but less amount of dedicated CPU time.

Last but not the least, since Node.js uses JavaScript internally, so it fits best for building client-side applications that also use JavaScript.

When To Not Use Node.Js?

However, we can use Node.js for a variety of applications. But it is a single threaded framework, so we should not use it for cases where the application requires long processing time. If the server is doing some calculation, it won’t be able to process any other requests. Hence, Node.js is best when processing needs less dedicated CPU time.

Q-4. What IDEs Can You Use For Node.Js Development?

**Answer.**

Here is the list of most commonly used IDEs for developing node.js applications.

Cloud9.

It is a free, cloud-based IDE that supports, application development, using popular programming languages like Node.js, PHP, C++, Meteor and more. It provides a powerful online code editor that enables a developer to write, run and debug the app code.

JetBrains WebStorm.

WebStorm is a lightweight yet powerful JavaScript IDE, perfectly equipped for doing client-side and server-side development using Node.js. The IDE provides features like intelligent code completion, navigation, automated and safe refactorings. Additionally, we can use the debugger, VCS, terminal and other tools present in the IDE.

JetBrains InteliJ IDEA.

It is a robust IDE that supports web application development using mainstream technologies like Node.js, Angular.js, JavaScript, HTML5 and more. To enable the IDE that can do Node.js development we have to install a Node.js plugin. It provides features, including syntax highlighting, code assistance, code completion and more. We can even run and debug Node.js apps and see the results right in the IDE. It’s JavaScript debugger offers conditional breakpoints, expression evaluation, and other features.

Komodo IDE.

It is a cross-platform IDE that supports development in main programming languages, like Node.js, Ruby, PHP, JavaScript and more. It offers a variety of features, including syntax highlighting, keyboard shortcuts, collapsible Pane, workspace, auto indenting, code folding and code preview using built-in browser.

Eclipse.

It is a popular cloud-based IDE for web development using Java, PHP, C++ and more. You can easily avail the features of Eclipse IDE using the Node.js plug-in, which is <**nodeclipse**>.

Atom.

It is an open source application built with the integration of HTML, JavaScript, CSS, and Node.js. It works on top of Electron framework to develop cross-platform apps using web technologies. Atom comes pre-installed with four UI and eight syntax themes in both dark and light colors. We can also install themes created by the Atom community or create our own if required.

Q-5. Explain How Does Node.Js Work?

**Answer.**

A Node.js application creates a single thread on its invocation. Whenever Node.js receives a request, it first completes its processing before moving on to the next request.

Node.js works asynchronously by using the event loop and callback functions, to handle multiple requests coming in parallel. An Event Loop is a functionality which handles and processes all your external events and just converts them to a callback function. It invokes all the event handlers at a proper time. Thus, lots of work is done on the back-end, while processing a single request, so that the new incoming request doesn’t have to wait if the processing is not complete.

While processing a request, Node.js attaches a callback function to it and moves it to the back-end. Now, whenever its response is ready, an event is called which triggers the associated callback function to send this response.

Let’s Take An Example Of A Grocery Delivery.

Usually, the delivery boy goes to each and every house to deliver the packet. Node.js works in the same way and processes one request at a time. The problem arises when any one house is not open. The delivery boy can’t stop at one house and wait till it gets opened up. What he will do next, is to call the owner and ask him to call when the house is open. Meanwhile, he is going to other places for delivery. Node.js works in the same way. It doesn’t wait for the processing of the request to complete (house is open). Instead, it attaches a callback function (call from the owner of the house) to it. Whenever the processing of a request completes (the house is open), an event gets called, which triggers the associated callback function to send the response.

To summarize, Node.js does not process the requests in parallel. Instead, all the back-end processes like, I/O operations, heavy computation tasks, that take a lot of time to execute, run in parallel with other requests.

Q-6. Explain REPL In Node.Js?

**Answer.**

The REPL stands for “Read Eval Print Loop”. It is a simple program that accepts the commands, evaluates them, and finally prints the results. REPL provides an environment similar to that of Unix/Linux shell or a window console, in which we can enter the command and the system, in turn, responds with the output. REPL performs the following tasks.

* **READ**
  + It Reads the input from the user, parses it into JavaScript data structure and then stores it in the memory.
* **EVAL**
  + It Executes the data structure.
* **PRINT**
  + It Prints the result obtained after evaluating the command.
* **LOOP**
  + It Loops the above command until the user presses Ctrl+C two times.

Q-7. Is Node.Js Entirely Based On A Single-Thread?

**Answer.**

Yes, it’s true that Node.js processes all requests on a single thread. But it’s just a part of the theory behind Node.js design. In fact, more than the single thread mechanism, it makes use of events and callbacks to handle a large no. of requests asynchronously.

Moreover, Node.js has an optimized design which utilizes both JavaScript and C++ to guarantee maximum performance. JavaScript executes at the server-side by Google Chrome v8 engine. And the C++ lib UV library takes care of the non-sequential I/O via background workers.

To explain it practically, let’s assume there are 100s of requests lined up in Node.js queue. As per design, the main thread of Node.js event loop will receive all of them and forwards to background workers for execution. Once the workers finish processing requests, the registered callbacks get notified on event loop thread to pass the result back to the user.

Q-8. How To Get Post Data In Node.Js?

**Answer.**

Following is the code snippet to fetch Post Data using Node.js.

app.use(express.bodyParser());

app.post('/', function(request, response){

console.log(request.body.user);

});

Q-9. How To Make Post Request In Node.Js?

**Answer.**

Following code snippet can be used to make a Post Request in Node.js.

var request = require('request');

request.post(

'http://www.example.com/action',

{ form: { key: 'value' } },

function (error, response, body) {

if (!error && response.statusCode == 200) {

console.log(body)

}

}

);

Q-10. What Is Callback In Node.Js?

**Answer.**

We may call “callback” as an asynchronous equivalent for a function. Node.js makes heavy use of callbacks and triggers it at the completion of a given task. All the APIs of Node.js are written in such a way that they support callbacks.

For example, suppose we have a function to read a file, as soon as it starts reading the file, Node.js return the control immediately to the execution environment so that the next instruction can be executed. Once file read operation is complete, it will call the callback function and pass the contents of the file as its arguments. Hence, there is no blocking or wait, due to File I/O. This functionality makes Node.js as highly scalable, using it processes a high number of requests without waiting for any function to return the expected result.

Q-11. What Is Callback Hell?

**Answer.**

Initially, you may praise Callback after learning about it. Callback hell is heavily nested callbacks which make the code unreadable and difficult to maintain.

Let’s see the following code example.

downloadPhoto('http://coolcats.com/cat.gif', displayPhoto)

function displayPhoto (error, photo) {

if (error) console.error('Download error!', error)

else console.log('Download finished', photo)

}

console.log('Download started')

In this scenario, Node.js first declares the “displayPhoto” function. After that, it calls the “downloadPhoto” function and pass the “displayPhoto” function as its callback. Finally, the code prints ‘Download started’ on the console. The “displayPhoto” will be executed only after “downloadPhoto” completes the execution of all its tasks.

Q-12. How To Avoid Callback Hell In Node.Js?

**Answer.**

Node.js internally uses a single-threaded event loop to process queued events. But this approach may lead to blocking the entire process if there is a task running longer than expected.

Node.js addresses this problem by incorporating callbacks also known as higher-order functions. So whenever a long-running process finishes its execution, it triggers the callback associated. With this approach, it can allow the code execution to continue past the long-running task.

However, the above solution looks extremely promising. But sometimes, it could lead to complex and unreadable code. More the no. of callbacks, longer the chain of returning callbacks would be. Just see the below example.

With such an unprecedented complexity, it’s hard to debug the code and can cause you a whole lot of time. There are four solutions which can address the callback hell problem.

1. Make Your Program Modular.

It proposes to split the logic into smaller modules. And then join them together from the main module to achieve the desired result.

2. Use Async Mechanism.

It is a widely used Node.js module which provides a sequential flow of execution.

The async module has <async.waterfall> API which passes data from one operation to other using the next callback.

Another async API <async.map> allows iterating over a list of items in parallel and calls back with another list of results.

With the async approach, the caller’s callback gets called only once. The caller here is the main method using the async module.

3. Use Promises Mechanism.

Promises give an alternate way to write async code. They either return the result of execution or the error/exception. Implementing promises requires the use of <.then()> function which waits for the promise object to return. It takes two optional arguments, both functions. Depending on the state of the promise only one of them will get called. The first function call proceeds if the promise gets fulfilled. However, if the promise gets rejected, then the second function will get called.

4. Use Generators.

Generators are lightweight routines, they make a function wait and resume via the yield keyword. Generator functions uses a special syntax <function\* ()>. They can also suspend and resume asynchronous operations using constructs such as promises or <thunks> and turn a synchronous code into asynchronous.

Q-13. Can You Create HTTP Server In Nodejs, Explain The Code Used For It?

**Answer.**

Yes, we can create HTTP Server in Node.js. We can use the <**http-server**> command to do so.

Following is the sample code.

var http = require('http');

var requestListener = function (request, response) {

response.writeHead(200, {'Content-Type': 'text/plain'});

response.end('Welcome Viewers\n');

}

var server = http.createServer(requestListener);

server.listen(8080); // The port where you want to start with.

Q-14. What Is The Difference Between Nodejs, AJAX, And JQuery?

**Answer.**

The one common trait between Node.js, AJAX, and jQuery is that all of them are the advanced implementation of JavaScript. However, they serve completely different purposes.

Node.Js –

It is a server-side platform for developing client-server applications. For example, if we’ve to build an online employee management system, then we won’t do it using client-side JS. But the Node.js can certainly do it as it runs on a server similar to Apache, Django not in a browser.

AJAX (Aka Asynchronous Javascript And XML) –

It is a client-side scripting technique, primarily designed for rendering the contents of a page without refreshing it. There are a no. of large companies utilizing AJAX such as Facebook and Stack Overflow to display dynamic content.

JQuery –

It is a famous JavaScript module which complements AJAX, DOM traversal, looping and so on. This library provides many useful functions to help in JavaScript development. However, it’s not mandatory to use it but as it also manages cross-browser compatibility, so can help you produce highly maintainable web applications.

Q-15. What Are Globals In Node.Js?

**Answer.**

There are three keywords in Node.js which constitute as Globals. These are Global, Process, and Buffer.

Global.

The Global keyword represents the global namespace object. It acts as a container for all other <global> objects. If we type <console.log(global)>, it’ll print out all of them.

An important point to note about the global objects is that not all of them are in the global scope, some of them fall in the module scope. So, it’s wise to declare them without using the var keyword or add them to Global object.

Variables declared using the var keyword become local to the module whereas those declared without it get subscribed to the global object.

Process.

It is also one of the global objects but includes additional functionality to turn a synchronous function into an async callback. There is no boundation to access it from anywhere in the code. It is the instance of the EventEmitter class. And each node application object is an instance of the Process object.

It primarily gives back the information about the application or the environment.

* **<process.execPath>** – to get the execution path of the Node app.
* **<process.Version> –** to get the Node version currently running.
* **<process.platform> –** to get the server platform.

Some of the other useful Process methods are as follows.

* **<process.memoryUsage> –** To know the memory used by Node application.
* **<process.NextTick> –** To attach a callback function that will get called during the next loop. It can cause a delay in executing a function.

Buffer.

The Buffer is a class in Node.js to handle binary data. It is similar to a list of integers but stores as a raw memory outside the V8 heap.

We can convert JavaScript string objects into Buffers. But it requires mentioning the encoding type explicitly.

* **<ascii> –** Specifies 7-bit ASCII data.
* **<utf8> –** Represents multibyte encoded Unicode char set.
* **<utf16le> –** Indicates 2 or 4 bytes, little endian encoded Unicode chars.
* **<base64> –** Used for Base64 string encoding.
* **<hex> –** Encodes each byte as two hexadecimal chars.

Here is the syntax to use the Buffer class.

> var buffer = new Buffer(string, [encoding]);

The above command will allocate a new buffer holding the string with <utf8> as the default encoding. However, if you like to write a <string> to an existing buffer object, then use the following line of code.

> buffer.write(string)

This class also offers other methods like <readInt8> and <writeUInt8> that allows read/write from various types of data to the buffer.

Q-16. How To Load HTML In Node.Js?

**Answer.**

To load HTML in Node.js we have to change the “Content-type” in the HTML code from text/plain to text/html.  
Let’s see an example where we have created a static file in web server.

fs.readFile(filename, "binary", function(err, file) {

if(err) {

response.writeHead(500, {"Content-Type": "text/plain"});

response.write(err + "\n");

response.end();

return;

}

response.writeHead(200);

response.write(file, "binary");

response.end();

});

Now we will modify this code to load an HTML page instead of plain text.

fs.readFile(filename, "binary", function(err, file) {

if(err) {

response.writeHead(500, {"Content-Type": "text/html"});

response.write(err + "\n");

response.end();

return;

}

response.writeHead(200, {"Content-Type": "text/html"});

response.write(file);

response.end();

});

Q-17. What Is EventEmitter In Node.Js?

**Answer.**

Events module in Node.js allows us to create and handle custom events. The Event module contains “EventEmitter” class which can be used to raise and handle custom events. It is accessible via the following code.

// Import events module

var events = require('events');

// Create an eventEmitter object

var eventEmitter = new events.EventEmitter();

When an EventEmitter instance encounters an error, it emits an “error” event. When a new listener gets added, it fires a “newListener” event and when a listener gets removed, it fires a “removeListener” event.

EventEmitter provides multiple properties like “on” and “emit”. The “on” property is used to bind a function to the event and “emit” is used to fire an event.

Q-18. How Many Types Of Streams Are Present In Node.Js?

**Answer.**

Stream in Node.js are objects that allow reading data from a source or writing data to a specific destination in a continuous fashion. In Node.js, there are four types of streams.

* **<Readable> –** This is the Stream to be used for reading operation.
* **<Writable> –** It facilitates the write operation.
* **<Duplex> –** This Stream can be used for both the read and write operations.
* **<Transform> –** It is a form of a duplex Stream, which performs the computations based on the available input.

All the Streams, discussed above are an instance of an “EventEmitter” class. The event thrown by the Stream varies with time. Some of the commonly used events are as follows.

* **<data> –** This event gets fired when there is data available for reading.
* **<end> –** The Stream fires this event when there is no more data to read.
* **<error> –** This event gets fired when there is any error in reading or writing data.
* **<finish> –** It fires this event after it has flushed all the data to the underlying system.

Q-19. List And Explain The Important REPL Commands?

**Answer.**

Following is the list of some of the most commonly used REPL commands.

* **<.help> –** It displays help for all the commands.
* **<tab Keys> –** It displays the list of all the available commands.
* **<Up/Down Keys> –** Its use is to determine what command was executed in REPL previously.
* **<.save filename> –** Save the current REPL session to a file.
* **<.load filename> –** To Load the specified file in the current REPL session.
* **<ctrl + c> –** used to Terminate the current command.
* **<ctrl + c (twice)> –** To Exit from the REPL.
* **<ctrl + d> –** This command perfoms Exit from the REPL.
* **<.break> –** It leads Exitting from multiline expression.
* **<.clear> –** Exit from multiline expression.

Q-20. What Is NPM In Node.Js?

**Answer.**

NPM stands for Node Package Manager. It provides following two main functionalities.

* It works as an Online repository for node.js packages/modules which are present at <nodejs.org>.
* It works as Command line utility to install packages, do version management and dependency management of Node.js packages.

NPM comes bundled along with Node.js installable. We can verify its version using the following command-

$ npm --version

NPM helps to install any Node.js module using the following command.

$ npm install <Module Name>

For example, following is the command to install a famous Node.js web framework module called express-

$ npm install express

Q-21. What Is The Global Installation Of Dependencies?

**Answer.**

Globally installed packages/dependencies are stored in <user-directory>/npm directory. Such dependencies can be used in CLI (Command Line Interface) function of any node.js, but cannot be imported using require() in the Node application directly.

To install a Node project globally use -g flag as.

C:\Nodejs\_WorkSpace>npm install express -g

Q-22. What Is The Local Installation Of Dependencies?

**Answer.**

By default, NPM installs any dependency in the local mode. It means that the package gets installed in “node\_modules” directory which is present in the same folder, where Node application is placed. Locally deployed packages are accessible via require(). Following is the syntax to install a Node project locally.

C:\Nodejs\_WorkSpace>npm install express

Q-23. What Is Package.Json? Who Uses It?

**Answer.**

What Is <Package.Json>?

* It is a plain JSON (JavaScript Object Notation) text file which contains all metadata information about Node.js Project or application.
* This file should be present in the root directory of every Node.js Package or Module to describe its metadata in JSON format.
* The file is named as “package” because Node.js platform treats every feature as a separate component. Node.js calls these as Package or Module.

Who Use It?

* NPM (Node Package Manager) uses <package.json> file. It includes details of the Node.js application or package. This file contains a no. of different directives or elements. These directives guide NPM, about how to handle a module or package.

Q-24. Does Node.Js Support Multi-Core Platforms? And Is It Capable Of Utilizing All The Cores?

**Answer.**

Yes, Node.js would run on a multi-core system without any issue. But it is by default a single-threaded application, so it can’t completely utilize the multi-core system.

However, Node.js can facilitate deployment on multi-core systems where it does use the additional hardware. It packages with a Cluster module which is capable of starting multiple Node.js worker processes that will share the same port.

Q-25. Which Is The First Argument Usually Passed To A Node.Js Callback Handler?

**Answer.**

Node.js core modules follow a standard signature for its callback handlers and usually the first argument is an optional error object. And if there is no error, then the argument defaults to null or undefined.

Here is a sample signature for the Node.js callback handler.

function callback(error, results) {

// Check for errors before handling results.

if ( error ) {

// Handle error and return.

}

// No error, continue with callback handling.

}

Q-26. What Is Chaining Process In Node.Js?

**Answer.**

It’s an approach to connect the output of one stream to the input of another stream, thus creating a chain of multiple stream operations.

Q-27. How To Create A Custom Directive In AngularJS?

**Answer.**

To create a custom directive, we have to first register it with the application object by calling the <directive> function. While invoking the <register> method of <directive>, we need to give the name of the function implementing the logic for that directive.

For example, in the below code, we have created a copyright directive which returns a copyright text.

app.directive('myCopyRight', function ()

{

return

{

template: '@CopyRight MyDomain.com '

};

});

**Note –** A custom directive should follow the camel case format as shown above.

Q-28. What Is A Child\_process Module In Node.Js?

**Answer.**

Node.js supports the creation of child processes to help in parallel processing along with the event-driven model.

The Child processes always have three streams <child.stdin>, child.stdout, and child.stderr. The <stdio> stream of the parent process shares the streams of the child process.

Node.js provides a <child\_process> module which supports following three methods to create a child process.

* **exec – <child\_process.exec>** method runs a command in a shell/console and buffers the output.
* **spawn – <child\_process.spawn>** launches a new process with a given command.
* **fork – <child\_process.fork>** is a special case of the spawn() method to create child processes.

Q-29. What Are The Different Custom Directive Types In AngularJS?

**Answer.**

AngularJS supports a no. of different directives which also depend on the level we want to restrict them.

So in all, there are four different kinds of custom directives.

* Element Directives (E)
* Attribute Directives (A)
* CSS Class Directives (C)
* Comment Directives (M)

Q-30. What Is A Control Flow Function? What Are The Steps Does It Execute?

**Answer.**

It is a generic piece of code which runs in between several asynchronous function calls is known as control flow function.

It executes the following steps.

* Control the order of execution.
* Collect data.
* Limit concurrency.
* Call the next step in the program.

**1)      What is node.js?**

Node.js is a Server side scripting which is used to build scalable programs. Its multiple advantages over other server side languages, the prominent being non-blocking I/O.

**2)      How node.js works?**

Node.js works on a v8 environment, it is a virtual machine that utilizes JavaScript as its scripting language and achieves high output via non-blocking I/O and single threaded event loop.

**3)      What do you mean by the term I/O ?**

I/O is the shorthand for input and output, and it will access anything outside of your application. It will be loaded into the machine memory to run the program, once the application is started.

[](https://career.guru99.com/wp-content/uploads/2014/03/NodeJS.png)

**4)      What does event-driven programming mean?**

In computer programming, event driven programming is a programming paradigm in which the flow of the program is determined by events like messages from other programs or threads. It is an application architecture technique divided into two sections 1) Event Selection 2) Event Handling

**5)      Where can we use node.js?**

Node.js can be used for the following purposes

a)      Web applications ( especially real-time web apps )

b)      Network applications

c)       Distributed systems

d)      General purpose applications

**6)      What is the advantage of using node.js?**

a)      It provides an easy way to build scalable network programs

b)      Generally fast

c)       Great concurrency

d)      Asynchronous everything

e)      Almost never blocks

**7)      What are the two types of API functions in Node.js ?**

The two types of API functions in Node.js are

a)      Asynchronous, non-blocking functions

b)      Synchronous, blocking functions

**8)      What is control flow function?**

A generic piece of code which runs in between several asynchronous function calls is known as control flow function.

**9)      Explain the steps how “Control Flow” controls the functions calls?**

a)      Control the order of execution

b)      Collect data

c)       Limit concurrency

d)      Call the next step in program

**10)   Why Node.js is single threaded**?

For async processing, Node.js was created explicitly as an experiment. It is believed that more performance and scalability can be achieved by doing async processing on a single thread under typical web loads than the typical thread based implementation.

**11)   Does node run on windows?**

Yes – it does. Download the MSI installer from <http://nodejs.org/download/>

**12)   Can you access DOM in node?**

No, you cannot access DOM in node.

**13)   Using the event loop what are the tasks that should be done asynchronously?**

a)      I/O operations

b)      Heavy computation

c)       Anything requiring blocking

**14)   Why node.js is quickly gaining attention from JAVA programmers?**

Node.js is quickly gaining attention as it is a loop based server for JavaScript. Node.js gives user the ability to write the JavaScript on the server, which has access to things like HTTP stack, file I/O, TCP and databases.

**15)   What are the two arguments that async.queue takes?**

The two arguments that async.queue takes

a)      Task function

b)      Concurrency value

**16)   What is an event loop in Node.js ?**

To process and handle external events and to convert them into callback invocations an event loop is used. So, at I/O calls, node.js can switch from one request to another .

**17)   Mention the steps by which you can async in Node.js?**

By following steps you can async Node.js

a)      First class functions

b)      Function composition

c)       Callback Counters

d)      Event loops

**18)    What are the pros and cons of Node.js?**

**Pros:**

a)      If your application does not have any CPU intensive computation, you can build it in Javascript top to bottom, even down to the database level if you use JSON storage object DB like MongoDB.

b)      Crawlers receive a full-rendered HTML response, which is far more SEO friendly rather than a single page application or a websockets app run on top of Node.js.

**Cons:**

a)       Any intensive CPU computation will block node.js responsiveness, so a threaded platform is a better approach.  
b)      Using relational database with Node.js is considered less favourable

**19)   How Node.js overcomes the problem of blocking of I/O operations?**

Node.js solves this problem by putting the event based model at its core, using an event loop instead of threads.

**20)   What is the difference between Node.js vs Ajax?**

The difference between Node.js and Ajax is that, Ajax (short for Asynchronous Javascript and XML) is a client side technology, often used for updating the contents of the page without refreshing it. While,Node.js is Server Side Javascript, used for developing server software. Node.js does not execute in the browser but by the server.

**21)   What are the Challenges with Node.js ?**

Emphasizing on the technical side, it’s a bit of challenge in Node.js to have one process with one thread to scale up on multi core server.

**22)** **What does it mean “non-blocking” in node.js?**

In node.js “non-blocking” means that its IO is non-blocking.  Node uses “libuv” to handle its IO in a platform-agnostic way. On windows, it uses completion ports for unix it uses epoll or kqueue etc. So, it makes a non-blocking request and upon a request, it queues it within the event loop which call the JavaScript ‘callback’ on the main JavaScript thread.

**23)   What is the command that is used in node.js to import external libraries?**

Command “require” is used for importing external libraries, for example, “var http=require (“http”)”.  This will load the http library and the single exported object through the http variable.

**24)   Mention the framework most commonly used in node.js?**

“Express” is the most common framework used in node.js

**25)   What is ‘Callback’ in node.js?**

Callback function is used in node.js to deal with multiple requests made to the server. Like if you have a large file which is going to take a long time for a server to read and if you don’t want a server to get engage in reading that large file while dealing with other requests, call back function is used. Call back function allows the server to deal with pending request first and call a function when it is finished.

**Node.js Interview Questions – Beginners Level**

**1. Differentiate between JavaScript and Node.js.**

|  |  |  |
| --- | --- | --- |
| ***Features*** | **JavaScript** | **Node.js** |
| *Type* | Programming Language | Interpreter and environment for JavaScript |
| *Utility* | Used for any client-side activity for a web application | Used for accessing or performing any non-blocking operation of any operating system |
| *Running Engine* | Spider monkey (FireFox), JavaScript Core (Safari), V8 (Google Chrome), etc. | V8 (Google Chrome) |

**2. What Is Node.js?**

Node.js is a powerful framework developed on **Chrome’s V8 JavaScript engine** that compiles the JavaScript directly into the native machine code. It is a lightweight framework used for creating server-side web applications and extends JavaScript API to offer usual server-side functionalities. It is generally used for large-scale application development, especially for video streaming sites, single page application, and other web applications.

**3. List down the major benefits of using Node.js?**

|  |  |
| --- | --- |
| **Features** | **Description** |
| ***Fast*** | Node.js is built on Google Chrome’s V8 JavaScript Engine which makes its library very fast in code execution |
| ***Asynchronous*** | Node.js based server never waits for an API to return data thus making it asynchronous |
| ***Scalable*** | It is highly scalable because of its event mechanism which helps the server to respond in a non-blocking way |
| ***Open Source*** | Node.js has an extensive open source community which has contributed in producing some excellent modules to add additional capabilities to Node.js applications |
| ***No Buffering*** | Node.js applications simply output the data in chunks and never buffer any data |

**4. What is the difference between Angular and Node.js?**

|  |  |
| --- | --- |
| **Angular** | **Node.js** |
| 1. It is an **open source** web application development framework | 1. It is a cross-platform **run-time environment** for applications |
| 2. It is written in TypeScript | 2. It is written in **C, C++ and JavaScript** languages |
| 3. Used for building **single-page** client-side web applications | 3. Used for building f**ast and scalable**server-side networking applications |
| 4. Angular itself is a **web application framework** | 4. Node.js has many **different frameworks**like Sails.js, Partial.js, and Express.js, etc. |
| 5. Ideal for creating **highly active and interactive** web apps | 5. Ideal for developing **small size**projects |
| 6. Helpful in **splitting an app into MVC components** | 6. Helpful in **generating database queries** |
| 7. Suitable for developing **real-time applications** | 7. Suitable in situations where something **faster and more scalable** is required |

**5. Why Node.js is single threaded?**

Node.js uses a single threaded model in order to support async processing. With async processing, an application can perform better and is more scalable under web loads. Thus, Node.js makes use of a [single-threaded model](https://www.edureka.co/blog/nodejs-tutorial/#nodearchitecture) approach rather than typical thread-based implementation.

**6. How do Node.js works?**

Node.js is a virtual machine that uses JavaScript as its scripting language and runs on a v8 environment. It works on a single-threaded event loop and a non-blocking I/O which provides high rate as it can handle a higher number of concurrent requests. Also, by making use of the [‘HTTP’ module](https://www.edureka.co/blog/nodejs-tutorial/#http), Node.js can run on any stand-alone web server.

**7. Where Node.js can be used?**

Node.js can be used to develop:

* Real-Time Web Applications
* Network Applications
* Distributed Systems
* General Purpose Applications

**8. How many types of API functions are there in Node.js?**

There are two types of API functions in Node.js:

* Asynchronous, non-blocking functions
* Synchronous, blocking functions

9. **What is the difference between Asynchronous and Non-blocking?**

|  |  |
| --- | --- |
| **Asynchronous** | **Non-blocking** |
| Asynchronous means not synchronous. Using these we can make asynchronous HTTP requests that do not wait for the server to respond. These functions continue to respond to the request for which it has already received the server response. | Non-blocking functions are used in regards with I/O operations. They immediately respond with whatever data is available and keeps on running as per the requests. In case, any answer couldn’t be retrieved then the API returns immediately with an error. |

**10. What is package.json?**

The [**package.json file**](https://www.edureka.co/blog/nodejs-tutorial/#json) in Node.js is the heart of the entire application. It is basically the manifest file that contains the metadata of the project where we define the properties of a package.

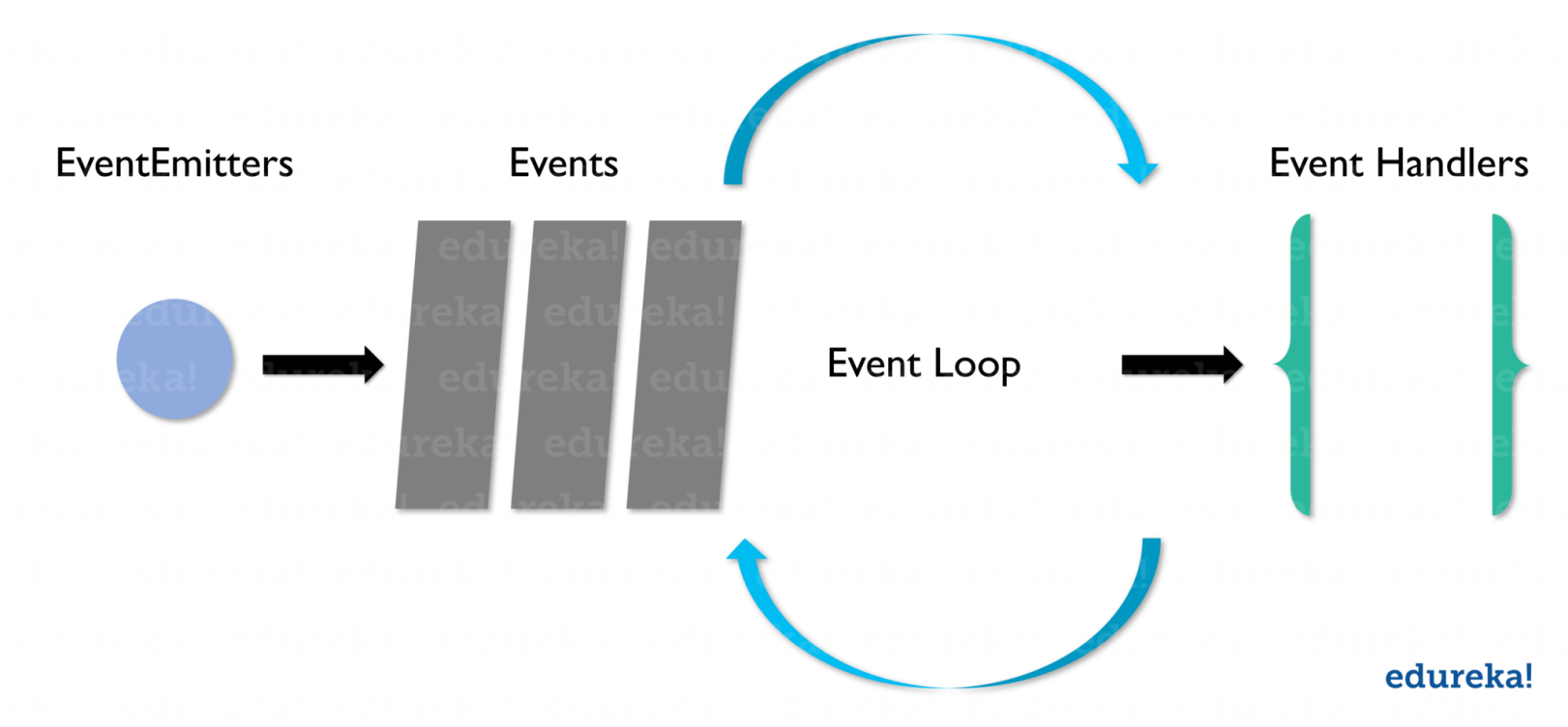


**11. What do you understand by Event-driven programming?**

Event-driven programming is a programming approach that heavily makes use of events for triggering various functions. An event can be anything like a mouse click, key press, etc. When an event occurs, a call back function is executed that is already registered with the element. This approach mainly follows the publish-subscribe pattern. Because of [event-driven programming](https://www.edureka.co/blog/nodejs-tutorial/#events), Node.js is faster when compared to other technologies.

**12. What is an *Event loop* in Node.js and how does it work?**

An event loop in Node.js handles all the asynchronous callbacks in an application. It is one of the most important aspects of Node.js and the reason behind Node.js have non-blocking I/O. Since Node.js is an event-driven language, you can easily attach a listener to an event and then when the event occurs the callback will be executed by the specific listener. Whenever functions like setTimeout, http.get, and fs.readFile are called, Node.js executed the event loop and then proceeds with the further code without waiting for the output. Once the entire operation is finished, Node.js receives the output and then executes the callback function. This is why all the callback functions are placed in a queue in a loop. Once the response is received, they are executed one by one.



**13. Explain  REPL in the context of Node.js.**

REPL in Node.js stands for **R**ead, **E**val, **P**rint, and **L**oop. It represents a computer environment such as a window console or Unix/Linux shell where any command can be entered and then the system can respond with an output. Node.js comes bundled with a REPL environment by default. REPL can perform the below-listed tasks:

[[](https://www.edureka.co/nodejs-certification-training)](https://www.edureka.co/nodejs-certification-training" \t "_blank)

**[Node.js Certification Training](https://www.edureka.co/nodejs-certification-training" \t "_blank)**

[Explore Curriculum](https://www.edureka.co/nodejs-certification-training" \t "_blank)

* **Read:** Reads the user’s input, parses it into JavaScript data-structure and then stores it in the memory.
* **Eval:** Receives and evaluates the data structure.
* **Print:**Prints the final result.
* **Loop:** Loops the provided command until *CTRL+C* is pressed twice.

**14. List down the tasks which should be done asynchronously using the event loop?**

Below is the list of the tasks which must be done asynchronously using the event loop:

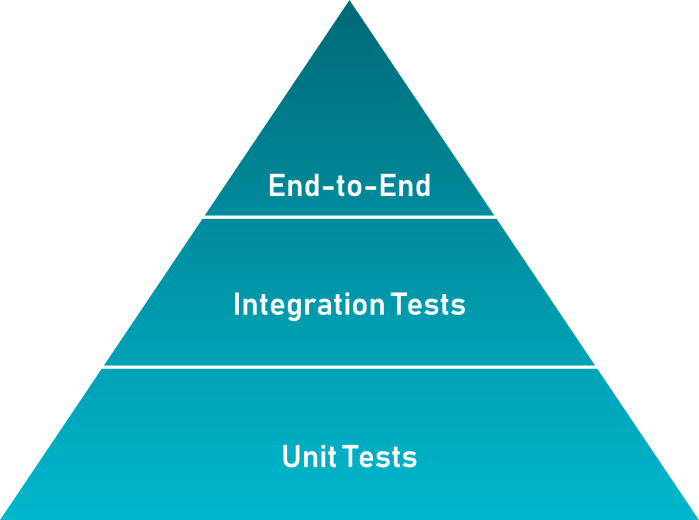
* I/O operations
* Heavy computation
* Anything requiring blocking

**15. List down the steps using which “Control Flow” controls the function calls in Node.js?**

1. Control the order of execution
2. Collect data
3. Limit concurrency
4. Call the next step in the program

**Node.js Interview Questions – Moderate Level**

**16. What do you understand by a test pyramid?**

A test pyramid basically is a diagram that describes the ratio of how many unit tests, integration tests, and end-to-end test are required to be written for the successful development of the project.

**17. What is an error-first callback in Node.js?**

Error-first callbacks in Node.js are used to pass errors and data. The very first parameter you need to pass to these functions has to be an error object while the other parameters represent the associated data. Thus you can pass the error object for checking if anything is wrong and handle it. In case there is no issue, you can just go ahead and with the subsequent arguments.

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7 | var myPost = new Post({title: 'edureka'});  myPost.save(function(err,myInstance){  if(err){  //handle error and return  }  //go ahead with `myInstance`  }); |

**18. Explain the purpose of module.exports?**

A module in Node.js is used to encapsulate all the related codes into a single unit of code which can be interpreted by shifting all related functions into a single file. For example, suppose you have a file called greet.js that contains the two functions as shown below:

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7 | module.exports = {  greetInHindi: function(){  return "NAMASTE";  },  greetInKorean: function(){  return "ANNYEONGHASEYO";  }}; |

As you can see module.exports provide two functions which can be imported in another file using below code:

|  |  |
| --- | --- |
| 1  2  3 | var eduGreets = require ("./greet.js");  eduGreets.greetInHindi() //NAMASTE  eduGreets.greetInKorean() //ANNYEONGHASEYO |

**19. What do you understand by Reactor Pattern in Node.js?**

**Reactor Pattern** in Node.js is basically a concept of non-blocking I/O operations. This pattern provides a handler that is associated with each I/O operation and as soon as an I/O request is generated, it is then submitted to a *demultiplexer*. This demultiplexer is a notification interface which is capable of handling concurrency in non-blocking I/O mode. It also helps in collecting each and every request in the form of an event and then place each event in a queue. Thus resulting in the generation of the Event Queue. Simultaneously, we have our event loop which iterates the events present in the Event Queue.

**20. What’s the difference between ‘front-end’ and ‘back-end’ development?**

|  |  |
| --- | --- |
| **Front-end Development** | **Back-end Development** |
| 1. Uses mark up and web languages like HTML, CSS, [JavaScript](https://www.edureka.co/blog/what-is-javascript/) | 1. Uses programming and scripting languages like [Python](https://www.edureka.co/blog/python-tutorial/), [Ruby](https://www.edureka.co/blog/ruby-on-rails-tutorial/), Perl, etc. |
| 2. Based on asynchronous requests and AJAX | 2. Based on Server Architecture |
| 3. Better Accessibility | 3. Enhanced Security |
| 4. Used for SEO | 4. Used for Backup |

**21. What are LTS releases of Node.js?**

**LTS**stands **L**ong **T**erm **S**upport version of Node.js that receives all the critical bug fixes along with security updates and performance improvements. These versions are supported for at least 18 months and mainly focus on stability and security. The modifications done to the LTS versions are restricted to the bug fixes, security upgrade, npm, and documentation updates, performance improvement, etc.

**22. List down the major security implementations within Node.js?**

Major security implementations in Node.js are:

1. Authentications
2. Error Handling

**23. What do you understand by callback hell?**

Callback Hell is also known as the Pyramid of Doom. It is a pattern caused by intensively nested callbacks which are unreadable and unwieldy. It typically contains multiple nested callback functions which in turn make the code hard to read and debug. It is caused by improper implementation of the asynchronous logic.

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9 | async\_A(function(){  async\_B(function(){  async\_C(function(){  async\_D(function(){  ....  });  });  });  }); |

**24. Explain libuv.**

Libuv is a multi-platform support library of Node.js which majorly is used for asynchronous I/O. It was primarily developed for Node.js,  with time it is popularly practiced with other systems like as Luvit, pyuv, Julia, etc. Libuv is basically an abstraction around libev/ IOCP depending on the platform, providing users an API based on libev. A few of the important features of libuv are:

* Full-featured event loop backed
* File system events
* Asynchronous file & file system operations
* Asynchronous TCP & UDP sockets
* Child processes

**25. Explain the concept of middleware in Node.js?**

In general, middleware is a function receives the Request and Response objects. In other words, in an application’s request-response cycle these functions have access to various request &  response objects along with the next function of the cycle. The next function of middleware is represented with the help of a variable, usually named next. Most commonly performed tasks by the middleware functions are:

* Execute any type of code
* Update or modify the request and the response objects
* Finish the request-response cycle
* Invoke the next middleware in the stack

**26. Explain the concept of URL module.**

The **URL module** of Node.js provides various utilities for **URL** resolution and parsing. It is a built-in module that helps in splitting up the web address into a readable format:

|  |  |
| --- | --- |
| 1 | var url = require('url'); |

**For example:**

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8 | var url = require('url');  var adrs = '<http://localhost:8082/default.htm?year=2019&month=april>';  var q = url.parse(adr, true);  console.log(q.host); //returns 'localhost:8082'  console.log(q.pathname); //returns '/default.htm'  console.log(q.search); //returns '?year=2019 and month=april'  var qdata = q.query; //returns an object: { year: 2019, month: 'april' }  console.log(qdata.month); //returns 'april' |

**27. What do you understand by ESLint?**

ESLint is an open source project initially developed by Nicholas C. Zakas in 2013 which aims to provide a linting utility for JavaScript through a plug. Linters in Node.js are good tools for searching certain bug classes, especially those which are related to the variable scope.

**28. For Node.js, why Google uses V8 engine?**

Google uses V8 as it is a Chrome runtime engine that converts JavaScript code into native machine code. This, in turn, speeds up the application execution and response process and give you a fast running application.

[[](https://www.edureka.co/nodejs-certification-training)](https://www.edureka.co/nodejs-certification-training" \t "_blank)

**[Node.js Certification Training](https://www.edureka.co/nodejs-certification-training" \t "_blank)**

[Watch The Course Preview](https://www.edureka.co/nodejs-certification-training" \t "_blank)

**29. Explain the working of the control flow function.**

In Node.js, the control flow function is basically the code that is executed between the asynchronous function calls. Below are the steps that must be followed for executing it:

1. Firstly, the order of execution must be controlled.
2. Then, the required data need to be collected.
3. Next, the concurrency must be limited.
4. Once done, the next step of the program has to be invoked.

**30. List down the two arguments that async.queue takes as input?**

Below are the two arguments that async.queue takes as input:

1. Task Function
2. Concurrency Value

**31. Differentiate between spawn() and fork() methods in Node.js?**

In Node.js, the spawn() is used to launch a new process with the provided set of commands. This method doesn’t create a new V8 instance and just one copy of the node module is active on the processor. When your child process returns a large amount of data to the Node you can invoke this method.

*Syntax:*

|  |  |
| --- | --- |
| 1 | child\_process.spawn(command[, args][, options]) |

Whereas, the fork() in Node.js is a special instance of spawn() that executes a new instance of the V8 engine. This method simply means that multiple workers are running on a single Node code base for various task.

*Syntax:*

|  |  |
| --- | --- |
| 1 | child\_process.fork(modulePath[, args][, options]) |

**32. What do you understand by global objects in Node.js?**

In Node.js, Globals are the objects which are global in nature and are available in all the modules of the application. You can use these objects directly in your application, rather than having to include them explicitly. The global objects can be modules, functions, strings, object, etc. Moreover, some of these objects can be in the module scope instead of global scope.

**33. Explain the concept of stub in Node.js.**

In Node.js, stubs are basically the programs or functions that are used for stimulating the module or component behavior. During any test cases, stubs provide the canned answers of the functions.

**34. How assert works in Node.js?**

In Node.js, assert is used to write tests. It only provides feedback only when any of the running test cases fails. This module gives you a set of assertion tests which are then used for testing invariants. It is basically used internally by Node.js but using require(‘assert’) code, it can be used in other applications as well.

|  |  |
| --- | --- |
| 1  2  3  4  5  6 | var assert = require('assert');  function mul(a, b) {  return a \* b;  }  var result = mul(1,2);  assert( result === 2, 'one multiplied by two is two'); |

**35. Define the concept of the test pyramid. Explain the process to implement them in terms of HTTP APIs.**

The test**pyramid** is basically a concept that is developed by Mike Cohn. According to this, you should have a higher number of low-level unit **tests** as compared to high-level end-to-end **tests that**running through a GUI.

In terms of HTTP APIs it may be defined as:

* A higher number of low-level unit tests for each model
* Lesser integration tests to test model interactions
* Lesser acceptance tests for testing actual HTTP endpoints

**36. Explain the purpose of ExpressJS package?**

Express.js is a framework built on top of Node.js that facilitates the management of the flow of data between server and routes in the server-side applications.  It is a lightweight and flexible framework that provides a wide range of features required for the web as well as mobile application development. Express.js is developed on the middleware module of Node.js called ***connect***. The connect module further makes use of **http** module to communicate with Node.js. Thus, if you are working with any of the connect based middleware modules, then you can easily integrate with Express.js.

**37. Differentiate between process.nextTick() and setImmediate()?**

In Node.js, process.nextTick() and setImmediate(), both are functions of the Timers module which help in executing the code after a predefined period of time. But these functions differ in their execution. The process.nextTick function waits for the execution of action till the next pass around in the event loop or once the event loop is completed only then it will invoke the callback function. On the other hand, setImmediate() is used to execute a callback method on the next cycle of the event loop which eventually returns it to the event loop in order to execute the I/O operations.

**38. Explain the usage of a buffer class in Node.js?**

Buffer class in Node.js is used for storing the raw data in a similar manner of an array of integers. But it corresponds to a raw memory allocation that is located outside the V8 heap. It is a global class that is easily accessible can be accessed in an application without importing a buffer module. Buffer class is used because pure JavaScript is not compatible with binary data. So, when dealing with TCP streams or the file system, it’s necessary to handle octet streams.

**39. How does Node.js handle the child threads?**

In general, Node.js is a single threaded process and doesn’t expose the child threads or thread management methods. But you can still make use of the child threads using spawn() for some specific asynchronous I/O tasks which execute in the background and don’t usually execute any JS code or hinder with the main event loop in the application. If you still want to use the threading concept in your application you have to include a module called ChildProcess explicitly.

[[](https://www.edureka.co/nodejs-certification-training)](https://www.edureka.co/nodejs-certification-training" \t "_blank)

**[Node.js Certification Training](https://www.edureka.co/nodejs-certification-training" \t "_blank)**

[Weekday / Weekend BatchesSee Batch Details](https://www.edureka.co/nodejs-certification-training" \t "_blank)

**40. Explain stream in Node.js along with its various types.**

Streams in Node.js are the collection of data similar to arrays and strings. They are objects using which you can read data from a source or write data to a destination in a continuous manner. It might not be available at once and need not to have fit in the memory. These streams are especially useful for reading and processing a large set of data. In Node.js, there are four fundamental types of streams:

1. *Readable:*Used for reading large chunks of data from the source.
2. *Writeable:* Use for writing large chunks of data to the destination.
3. *Duplex:*Used for both the functions; read and write.
4. *Transform:*It is a duplex stream that is used for modifying the data.

**Node.js Interview Questions – Advanced Level**

**41. What is the use of NODE\_ENV?**

If the project is in the production stage, Node.js promotes the convention of making use of NODE\_ENV variable to flag it. This helps in taking better judgment during the development of the projects. Also, when you set your NODE\_ENV to production, your application tends to perform 3 times faster.

**42. Differentiate between readFile vs createReadStream in Node.js?**

Node.js provides two ways to read and execute files which are using readFile and CreateStream. readFile() is a fully buffered process which returns the response only when the complete file is pushed into the buffer and is read. It is a memory intensive process and in case of large files, the processing can be very slow. Whereas createReadStream is a partially buffered which treats the entire process as an event series. The entire file is split into chunks which are then processed and sent back as a response one by one. Once done, they are finally removed from the buffer. Unlike readFile, createReadStream is really effective for the processing of the large files.

**43. List down the various timing features of Node.js.**

Node.js provides a Timers module which contains various functions for executing the code after a specified period of time. Below I have listed down the various functions provided by this module:

* **setTimeout/clearTimeout** – Used to schedule code execution after a designated amount of milliseconds
* **setInterval/clearInterval** – Used to execute a block of code multiple times
* **setImmediate/clearImmediate** – Used to execute code at the end of the current event loop cycle
* **process.nextTick** – Used to schedule a callback function that needs to be invoked in the next iteration of the Event Loop

**44. Explain the concept of Punycode in Node.js?**

In Node.js, Punycode is an encoding syntax that is used for converting Unicode (UTF-8) string of characters into a basic ASCII string of characters. It is important as the hostnames can only understand the ASCII characters. Thus, Node.js version 0.6.2 onwards, it was bundled up with the default Node package. If you want to use it with any previous versions, you can easily do that by using the following code:

Syntax:

|  |  |
| --- | --- |
| 1 | punycode = require('punycode'); |

**45. Differentiate between Node.js vs Ajax?**

The most basic difference between Node.js and Ajax that, Node.js is a server-side JavaScript whereas Ajax is a client-side technology. In simpler terms, Ajax is mostly used for updating or modifying the webpage contents without having to refresh it. On the other hand, Node.js is required to develop the server software that are typically executed by the servers instead of the web browsers.

**46. Does Node.js provide any Debugger?**

Node.js do provide a simple TCP based protocol and debugging client that comes built-in. In order to debug your JavaScript file, you can use the below debug argument followed by **js** file name that you want to debug.

*Syntax:*

|  |  |
| --- | --- |
| 1 | node debug [script.js | -e "script" | <host> : <port> ] |

**47. Describe the exit codes of Node.js.**

In Node.js, exit codes are a set of specific codes which are used for finishing a specific process. These processes can include the global object as well. Below are some of the exit codes used in Node.js:

* Uncaught fatal exception
* Unused
* Fatal Error
* Internal Exception handler Run-time failure
* Internal JavaScript Evaluation Failure

**48. What do you understand by an Event Emitter in Node.js?**

EventEmitter is a Node.js class that includes all the objects that are capable of emitting events. These objects contain an eventEmitter.on() function through which more than one function can be attached to the named events that are emitted by the object. Whenever an EventEmitter object throws an event, all the attached functions to that specific event are invoked synchronously. Below code shows how to us the EventEmitter in your application:

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7 | const EventEmitter = require('events');  class MyEmitter extends EventEmitter { }  const myEmitter = new MyEmitter();  myEmitter.on('event', () => {  console.log('an event occurred!');  });  myEmitter.emit('event'); |

**49. Is cryptography supported in Node.js?**

Yes, Node.js does support cryptography through a module called Crypto. This module provides various cryptographic functionalities like cipher, decipher, sign and verify functions, a set of wrappers for open SSL’s hash HMAC etc. For example:

*Syntax:*

|  |  |
| --- | --- |
| 1  2  3  4 | const crypto = require'crypto');  const secret = 'akerude';  const hash = crypto.createHmac('swaEdu', secret).update('Welcome to Edureka').digest('hex');  console.log(hash); |

**50. Explain the reason as to why Express ‘app’ and ‘server’ must be kept separate.**

Express ‘app’ and ‘server’ must be kept separate as by doing this, you will be separating the API declaration from the network related configuration which benefits in the below listed ways:

* It allows testing the API in-process without having to perform the network calls
* Faster testing execution
* Getting wider coverage metrics of the code
* Allows deploying the same API under flexible and different network conditions
* Better separation of concerns and cleaner code

API declaration should reside in app.js:

|  |  |
| --- | --- |
| 1  2  3  4 | var app = express();  app.use(bodyParser.json());  app.use("/api/events", events.API);  app.use("/api/forms", forms); |

Server network declaration should reside in /bin/www:

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7 | var app = require('../app');  var http = require('http');  //Get port from environment and store in Express  var port = normalizePort(process.env.PORT || '8000');  app.set('port', port);  //Create HTTP server.  var server = http.createServer(app); |

Node.js Interview Questions

A list of top frequently asked **Node.js interview questions** and answers are given below.

1) What is Node.js?

Node.js is Server-side scripting which is used to build scalable programs. It is a web application framework built on Google Chrome's JavaScript Engine. It runs within the Node.js runtime on Mac OS, Windows, and Linux with no changes. This runtime facilitates you to execute a JavaScript code on any machine outside a browser.

2) Is Node.js free to use?

Yes. It is released under MIT license and is free to use.

3) Is Node a single threaded application?

Yes. Node is a single-threaded application with event looping.

4) What is the purpose of Node.js?

These are the following purposes of Node.js:

* Real-time web applications
* Network applications
* Distributed systems
* General purpose applications

5) What are the advantages of Node.js?

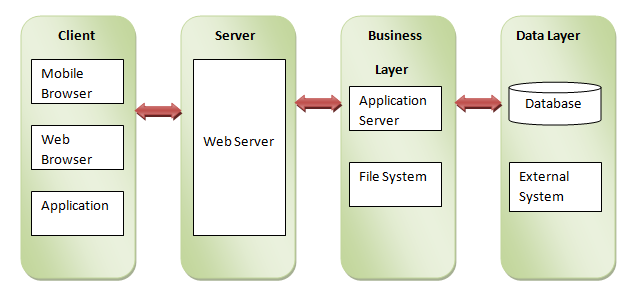
Following are the main advantages of Node.js:

* Node.js is asynchronous and event-driven. All API?s of Node.js library are non-blocking, and its server doesn't wait for an API to return data. It moves to the next API after calling it, and a notification mechanism of Events of Node.js responds to the server from the previous API call.
* Node.js is very fast because it builds on Google Chrome?s V8 JavaScript engine. Its library is very fast in code execution.
* Node.js is single threaded but highly scalable.
* Node.js provides a facility of no buffering. Its application never buffers any data. It outputs the data in chunks.

6) Explain Node.js web application architecture?

A web application distinguishes into 4 layers:

* **Client Layer:**The Client layer contains web browsers, mobile browsers or applications which can make an HTTP request to the web server.
* **Server Layer:**The Server layer contains the Web server which can intercept the request made by clients and pass them the response.
* **Business Layer:**The business layer contains application server which is utilized by the web server to do required processing. This layer interacts with the data layer via database or some external programs.
* **Data Layer:**The Data layer contains databases or any source of data.



7) What do you understand by the term I/O?

I/O stands for input and output. It accesses anything outside of your application. It loaded into the machine memory to run the program, once the application starts.

8) How many types of API functions are available in Node.js?

There are two types of API functions in Node.js:

* Asynchronous, Non-blocking functions
* Synchronous, Blocking functions

9) What is error-first callback?

Error-first callbacks are used to pass errors and data. If something goes wrong, the programmer has to check the first argument because it is always an error argument. Additional arguments are used to pass data.

1. fs.readFile(filePath, function(err, data) {
2. if (err) {
3. //handle the error
4. }
5. // use the data object
6. });

10) What is an asynchronous API?

All the API's of Node.js library are asynchronous means non-blocking. A Node.js based server never waits for an API to return data. The Node.js server moves to the next API after calling it, and a notification mechanism of Events of Node.js responds to the server for the previous API call.

11) How can you avoid callbacks?

To avoid callbacks, you can use any one of the following options:

* You can use **modularization**. It breaks callbacks into independent functions.
* You can use **promises**.
* You can use **yield** with Generators and Promises.

12) Does Node.js provide Debugger?

Yes, Node.js provides a simple TCP based protocol and built-in debugging client. For debugging your JavaScript file, you can use debug argument followed by the js file name you want to debug.

Syntax:

1. node debug [script.js | -e "script" | **<host>**:**<port>**]

13) What is a control flow function?

Control flow function is a generic piece of code that runs in between several asynchronous function calls.

14) How "Control Flow" controls the functions calls?

The control flow does the following job:

* Control the order of execution
* Collect data
* Limit concurrency
* Call the next step in a program

15) Is it possible to access DOM in Node?

No, it is not possible to access DOM in Node.

16) What types of tasks can be done asynchronously using the event loop?

* I/O operations
* Heavy computation
* Anything requiring blocking

17) What is REPL in Node.js?

REPL stands for Read Eval Print Loop. It specifies a computer environment like a window console or Unix/Linux shell where you can enter a command, and the computer responds with an output.

REPL environment incorporates with Node.js.

18) Explain the tasks of terms used in Node REPL.

Following are the terms used in REPL with their defined tasks:

**Read:** It reads user's input; parse the input into JavaScript data-structure and stores in memory.

**Eval:** It takes and evaluates the data structure.

**Print:** It is used to print the result.

**Loop:** It loops the above command until user press ctrl-c twice to terminate.

19) Is it possible to evaluate simple expressions using Node REPL?

Yes. You can evaluate simple expressions using Node REPL.

20) What is the use of the underscore variable in REPL?

In REPL, the underscore variable is used to get the last result.

1. C:\Nodejs\_WorkSpace**>**node
2. **>** var x = 10
3. undefined
4. **>** var y = 20
5. undefined
6. **>** x + y
7. 30
8. **>** var sum = \_
9. undefined
10. **>** console.log(sum)
11. 30
12. undefined
13. **>**

21) Does Node.js supports cryptography?

Yes, Node.js Crypto module supports cryptography. It provides cryptographic functionality that includes a set of wrappers for open SSL's hash HMAC, cipher, decipher, sign and verify functions. For example:

1. const crypto = require('crypto');
2. const secret = 'abcdefg';
3. const hash = crypto.createHmac('sha256', secret)
4. .update('Welcome to JavaTpoint')
5. .digest('hex');
6. console.log(hash);

22) What is npm? What is the main functionality of npm?

npm stands for Node Package Manager. Following are the two main functionalities of npm:

* Online repositories for node.js packages/modules which are searchable on search.nodejs.org
* Command line utility to install packages, do version management and dependency management of Node.js packages.

23) What tools can be used to assure a consistent style in Node.js?

Following is a list of tools that can be used in developing code in teams, to enforce a given style guide and to catch common errors using static analysis.

* JSLint
* JSHint
* ESLint
* JSCS

24) What is the difference between operational and programmer errors?

Operational errors are not bugs, but create problems with the system like request timeout or hardware failure. On the other hand, programmer errors are actual bugs.

25) What is the difference between the global installation of dependencies and local installation of dependencies?

* Global installation of dependencies is stored in /npm directory. While local installation of dependencies stores in the local mode. Here local mode refers to the package installation in node\_modules directory lying in the folder where Node application is present.
* Globally deployed packages cannot be imported using require() in Node application directly. On the other hand, locally deployed packages are accessible via require().
* To install a Node project globally -g flag is used.
  1. C:\Nodejs\_WorkSpace**>**npm install express ?g
* To install a Node project locally, the syntax is:
  1. C:\Nodejs\_WorkSpace**>**npm install express

26) What is the use of a buffer class in Node.js?

The Node.js provides Buffer class to store raw data similar to an array of integers but corresponds to a raw memory allocation outside the V8 heap. It is a global class and can be accessed in an application without importing a buffer module. Buffer class is used because pure JavaScript is not compatible with binary data. So, when dealing with TCP streams or the file system, it's necessary to handle octet streams.

27) What is the role of assert in Node.js?

The Node.js Assert is a way to write tests. It provides no feedback when running your test unless one fails. The assert module provides a simple set of assertion tests that can be used to test invariants. The module is intended for internal use by Node.js, but can be used in application code via require ('assert'). For example:

1. var assert = require('assert');
2. function add (a, b) {
3. return a + b;
4. }
5. var expected = add(1,2);
6. assert( expected === 3, 'one plus two is three');

28) What are the streams in Node.js?

The Streams are the objects that facilitate you to read data from a source and write data to a destination. There are four types of streams in Node.js:

* **Readable:**This stream is used for reading operations.
* **Writable:**This stream is used for write operations.
* **Duplex:**This stream can be used for both reading and write operations.
* **Transform:**It is a type of duplex stream where the output computes according to input.

29) What is event-driven programming in Node.js?

In Node.js, event-driven programming means as soon as Node starts its server, it initiates its variables, declares functions and then waits for an event to occur. It is one of the reasons why Node.js is pretty fast compared to other similar technologies.

**Part 1 – Express JS Interview Questions (Basic)**

This first part covers basic Express JS Interview Questions and answers

**1. What do you mean by Express JS and what is its use?**

 Popular Course in this category

**[JavaScript Certification Course](https://www.educba.com/software-development/courses/javascript-certification-course/?btnz=edu-blg-inline-banner1" \t "_blank)**[47 Online Courses | 210+ Hours | Verifiable Certificate of Completion | Lifetime Access   
4.5 (1,317 ratings)](https://www.educba.com/software-development/courses/javascript-certification-course/?btnz=edu-blg-inline-banner1" \t "_blank)

[Course Price   
₹12999](https://www.educba.com/software-development/courses/javascript-certification-course/?btnz=edu-blg-inline-banner1" \t "_blank)~~[₹33999](https://www.educba.com/software-development/courses/javascript-certification-course/?btnz=edu-blg-inline-banner1" \t "_blank)~~**[View Course](https://www.educba.com/software-development/courses/javascript-certification-course/?btnz=edu-blg-inline-banner1" \t "_blank)**

Related Courses

[Programming Languages Course](https://www.educba.com/software-development/courses/programming-languages-course/?btnz=edu-blg-inline-banner1)[C Programming Course](https://www.educba.com/software-development/courses/c-programming-course/?btnz=edu-blg-inline-banner1)[Selenium Training Certification](https://www.educba.com/software-development/courses/selenium-training-certification/?btnz=edu-blg-inline-banner1)

Answer:  
Express JS is an application framework that is light-weighted node JS. Variety of versatile, helpful and vital options are provided by this [JavaScript framework](https://www.educba.com/careers-in-javascript/) for the event of mobile additionally as internet applications with the assistance of node JS.  
Express JS Use – Express.js could be a light-weight internet application that helps in organizing the net application into MVC design on the server aspect.

**2. Write the steps for setting up an Express JS application?**

Answer:  
Following are the steps accustomed for An Express JS application: –

1. A folder with a constant name because the project name is made.
2. A file named package.json is made within the folder created.
3. “npm install” command is run on the electronic communication. It installs all the libraries gift in package.json.
4. A file named server.js is made.
5. “Router” file is made within the package that consists of a folder named index.js.
6. “App” is made within the package that has the index.html file.

Let us move on to the next Express JS Interview Questions

**3. What function are arguments available to Express JS route handlers?**

Answer:  
The arguments which are available to an Express JS route handler-function are-

• Req – the request object  
• Res – the response object  
• Next (optional) – a function that is employed to pass management to 1 of the following route handlers.

The third argument is optional and should be omitted, however, in some cases, it’s helpful wherever there’s a series of handlers and management will be passed to 1 of the following route handlers skipping this one.

**4.How to Config properties in Express JS?**

Answer:  
In Express JS, there are 2 ways that for configuring the properties:

With process.ENV:

* A file with the name “.env” is to be created within the project folder.
* All the properties are to be other within the “.env” file.
* Any of the properties will be employed in server.js.

With require.JS:

* A file with the name “config.json” is to be created within the config folder within the project folder.
* The config properties are to be present within the config.json file.
* Now, ought to be accustomed access the config.json file.

**5. How Should I Structure my Express JS Application?**

Answer:  
This is the basic Express JS Interview Questions asked in an interview. There is no definitive answer to the current question. The solution depends on the dimensions of your application and therefore the team that’s concerned.  
Routes and alternative application-Express logic will board as several files as you would like, in any directory structure you favor. Read the subsequent examples for inspiration:

* Route listings
* Route map
* MVC vogue controllers

Also, there are third-party extensions for Express JS applications that modify a number of these patterns:

* Resourceful routing

**Part 2 – Express JS Interview Questions (Advanced)**

Let us now have a look at the advanced Express JS Interview Questions.

**6. How to allow CORS in Express JS? Explain with an example?**

Answer:  
In order to permit CORS in Express.js, add the subsequent code in server.js:  
For Example –  
app.all(‘\*’, function(req, res, next) {  
res.set(‘Access-Control-Allow-Origin’, ‘\*’);  
res.set(‘Access-Control-Allow-Methods’, ‘GET, POST, DELETE, PUT’);  
res.set(‘Access-Control-Allow-Headers’, ‘X-Requested-With, Content-Type’);  
if (‘OPTIONS’ == req.method) return res.send(200);  
next();  
});

**7.How to enable debugging in express app?**

Answer:  
In different operative Systems, we’ve got following commands:

On UNIX operating system the command would be as follows:

* $ DEBUG=express:\* node index.js

On Windows the command would be:

* set DEBUG=express:\* & node index.js

From Webstrome IDE

* C:\Program Files (x86)\JetBrains\WebStorm 2016.2.4\bin\runnerw.exe” “C:\Program Files\nodejs\node.exe” –debug-brk=61081 –expose\_debug\_as=v8debug E:\Development\nodejd\librarey\bin\www

Let us move on to the next advanced Express JS Interview Questions.

**8. Explain Error Handling In Express.js Using An Example?**

Answer:  
From Express 4.0 Error handling is easier. The steps are as following:

Create a Middleware:  
•Create a middleware as following:  
// error handler  
app.use(function(err, req, res, next) solely providing error in development  
res.locals.message = err.message;  
res.locals.error = req.app.get(‘env’) === ‘development’ ? err : ;  
// render the error page  
res.status(err.status || 500);  
res.render(‘error’);  
});  
Install Error Handler Middleware:

• Install errorhandler.  
npm install errorhandler –save

• Create a vareiable.  
vare errorhandler = require(‘errorhandler’)

• Use the middleware as following:  
if (process.env.NODE\_ENV === ‘development’) solely use in development  
app.use(errorhandler())  
}  
function errorNotification(err, str, req) power unit title = ‘Error in ‘ + req.method + ‘ ‘ + req.url  
notifier.notify()  
}

**9.What is the use of next in Express JS?**

Answer:  
Next -It passes management to a consecutive matching route. OR a operate to pass management to 1 of the following route handlers.  
The argument could also be omitted, however, is beneficial in cases wherever you have got a series of handlers and you’d wish to pass management to 1 of the following route handlers, and skip this one.  
app.get(‘/user details/:id?’, function(req, res, next));  
Req and Res – It represents the request and response objects  
Next – It passes management to a consecutive matching route.

**10. How to Redirect 404 Errors to A Page In ExpressJS?**

Answer:  
In server.js add the subsequent code to send 404 errors back to a page in our ExpressJS App:  
/\* Define fallback route \*/  
app.use(function(req, res, next) {  
res.status(404).json({errorCode: 404, errorMsg: “route not found”});  
});

1. **Question 1. What Is Express Js?**

**Answer :**

Express JS is a framework which helps to develop web and mobile applications. Its works on nodejs plateform. Its sub part of node.js.

1. **Question 2. What Type Of Web Application Can Built Using Express Js?**

**Answer :**

you can build single-page, multi-page, and hybrid web applications.

[Java Script Interview Questions](https://www.wisdomjobs.com/e-university/java-script-interview-questions.html)

1. **Question 3. What Are Core Features Of Express Framework?**

**Answer :**

* 1. Allows to set up middlewares to respond to HTTP Requests
  2. Defines a routing table which can works as per HTTP Method and URL.
  3. Dynamically render HTML Pages

1. **Question 4. Why I Should Use Express Js?**

**Answer :**

Express 3.x is a light-weight web application framework to help organize your web application into an MVC architecture on the server side.

[Java Script Tutorial](https://www.wisdomjobs.com/e-university/java-script-tutorial-209.html" \o "Java Script Tutorial)

1. **Question 5. How To Install Expressjs?**

**Answer :**

Assuming you’ve already installed Node.js, create a directory to hold your application, and make that your working directory.

$ mkdir myapp  
$ cd myapp

Use the npm init command to create a package.json file for your application. For more information on how package.json works, see Specifics of npm’s package.json handling.

$ npm init

This command prompts you for a number of things, such as the name and version of your application. For now, you can simply hit RETURN to accept the defaults for most of them, with the following exception:

entry point: (index.js)

Enter app.js, or whatever you want the name of the main file to be. If you want it to be index.js, hit RETURN to accept the suggested default file name.

Now install Express in the myapp directory and save it in the dependencies list. For example:

$ npm install express --save

To install Express temporarily and not add it to the dependencies list, omit the --save option:

$ npm install express

[AJAX Interview Questions](https://www.wisdomjobs.com/e-university/ajax-interview-questions.html" \o "AJAX Interview Questions)

1. **Question 6. How To Get Variables In Express.js In Get Method?**

**Answer :**

var express = require('express');  
var app = express();  
app.get('/', function(req, res){  
    /\* req have all the values \*\*/    
  res.send('id: ' + req.query.id);    
});  
app.listen(3000);

1. **Question 7. How To Get Post A Query In Express.js?**

**Answer :**

var bodyParser = require('body-parser')  
app.use( bodyParser.json() );       // to support JSON-encoded   
app.use(bodyParser.urlencoded({     // to support URL-encoded   
  extended: true  
}));

[AJAX Tutorial](https://www.wisdomjobs.com/e-university/ajax-tutorial-267.html" \o "AJAX Tutorial) [MySQL Interview Questions](https://www.wisdomjobs.com/e-university/mysql-interview-questions.html" \o "MySQL Interview Questions)

1. **Question 8. How To Output Pretty Html In Express.js?**

**Answer :**

app.set('view options', { pretty: true });

1. **Question 9. How To Get The Full Url In Express?**

**Answer :**

var port = req.app.settings.port || cfg.port;

res.locals.requested\_url = req.protocol + '://' + req.host  + ( port == 80 || port == 443 ? '' : ':'+port ) + req.path;

[HTML+Javascript Interview Questions](https://www.wisdomjobs.com/e-university/html-javascript-interview-questions.html" \o "HTML+Javascript Interview Questions)

1. **Question 10. How To Remove Debugging From An Express App?**

**Answer :**

var io = require('socket.io').listen(app, { log: false });  
io.set('log level', 1);

[MySQL Tutorial](https://www.wisdomjobs.com/e-university/mysql-tutorial-275.html" \o "MySQL Tutorial)

1. **Question 11. How To Do 404 Errors?**

**Answer :**

app.get('\*', function(req, res){  
  res.send('what???', 404);  
});

[JavaServer Faces (JSF) Interview Questions](https://www.wisdomjobs.com/e-university/javaserver-faces-jsf-interview-questions.html" \o "JavaServer Faces (JSF) Interview Questions)

1. **Question 12. How To Download A File?**

**Answer :**

app.get('/download', function(req, res){  
  var file = \_\_dirname + '/download-folder/file.txt';  
  res.download(file);   
});

[Java Script Interview Questions](https://www.wisdomjobs.com/e-university/java-script-practice-tests-209-327258" \o "Java Script Interview Questions)

1. **Question 13. What Is The Parameter “next” Used For In Express?**

**Answer :**

app.get('/userdetails/:id?', function(req, res, next){  
 });

req and res which represent the request and response objects

nextIt passes control to the next matching route.

[JavaServer Faces (JSF) Tutorial](https://www.wisdomjobs.com/e-university/javaserver-faces-jsf-tutorial-1064.html" \o "JavaServer Faces (JSF) Tutorial)

1. **Question 14. What Function Arguments Are Available To Express.js Route Handlers?**

**Answer :**

The arguments available to an Express.js route handler function are:

* 1. req - the request object
  2. res - the response object
  3. next (optional) - a function to pass control to one of the subsequent route handlers

The third argument may be omitted, but is useful in cases where you have a chain of handlers and you would like to pass control to one of the subsequent route handlers, and skip the current one.

1. **Question 15. How To Config Properties In Express Application?**

**Answer :**

In an ExpressJS Application, we can config properties in following two ways:

**With Process.ENV:**

* 1. Create a file with name ‘.env’ inside the project folder.
  2. Add all the properties in ‘.env’ file.
  3. In server.js any of the properties can be used as:  
       
     var host = process.env.APP\_HOST   
     app.set('host', host);  
     logger.info('Express server listening on http://' + app.get('host'));

**With RequireJs:**

* 1. Create a file called  ‘config.json’ inside a folder called ‘config’ inside the project folder.
  2. Add config properties in config.json.  
       
     {  
      "env":"development",  "apiurl":"http://localhost:9080/api/v1/"  
     }
  3. Use require to access the config.json file.  
       
     var config = require('./config/config.json');

[Node.js Interview Questions](https://www.wisdomjobs.com/e-university/nodedotjs-interview-questions.html" \o "Node.js Interview Questions)

1. **Question 16. How To Allow Cors In Expressjs? Explain With An Example?**

**Answer :**

In order to allow CORS in Express.js,  add the following code in server.js:

app.all('\*', function(req, res, next) {  
res.set('Access-Control-Allow-Origin', '\*');  
res.set('Access-Control-Allow-Methods', 'GET, POST, DELETE, PUT');  
res.set('Access-Control-Allow-Headers', 'X-Requested-With, Content-Type');  
if ('OPTIONS' == req.method) return res.send(200);  
next();  
});

[Node.js Tutorial](https://www.wisdomjobs.com/e-university/nodedotjs-tutorial-1161.html" \o "Node.js Tutorial)

1. **Question 17. How To Redirect 404 Errors To A Page In Expressjs?**

**Answer :**

In server.js add the following code to redirect 404 errors back to a page in our ExpressJS App:

/\* Define fallback route \*/  
app.use(function(req, res, next) {  
    res.status(404).json({errorCode: 404, errorMsg: "route not found"});  
});

[CSS Interview Questions](https://www.wisdomjobs.com/e-university/css-interview-questions.html" \o "CSS Interview Questions)

1. **Question 18. Explain Error Handling In Express.js Using An Example?**

**Answer :**

From Express 4.0 Error handling is much easier. The steps are as following:

* 1. Create an express.js application and as there is no built-in middleware like errorhandler in express 4.0, therefore, the middleware need to be either installed or need to create a custom one.

**Create a Middleware:**

* 1. Create a middleware as following:  
       
     // error handler  
     app.use(function(err, req, res, next) {  
     // set locals, only providing error in development  
     res.locals.message = err.message;  
     res.locals.error = req.app.get('env') === 'development' ? err : {};  
     // render the error page  
     res.status(err.status || 500);  
     res.render('error');  
     });

**Install Error Handler Middleware:**

* 1. Install errorhandler.  
     npm install errorhandler --save
  2. Create a variable.  
     var errorhandler = require('errorhandler')
  3. Use the middleware as following:  
       
     if (process.env.NODE\_ENV === 'development') {  
      // only use in development   
       app.use(errorhandler({log: errorNotification}))  
     }  
     function errorNotification(err, str, req) {  
      var title = 'Error in ' + req.method + ' ' + req.url  
      notifier.notify({  
        title: title,  
        message: str  
      })  
     }

[AJAX Interview Questions](https://www.wisdomjobs.com/e-university/ajax-interview-questions.html" \o "AJAX Interview Questions)

1. **Question 19. How To Enable Debugging In Express App?**

**Answer :**

In different Operating Systems, we have following commands:

**On Linux the command would be as follows:**

  $ DEBUG=express:\* node index.js

**On Windows the command would be:**

  set DEBUG=express:\* & node index.js

[CSS Tutorial](https://www.wisdomjobs.com/e-university/css-tutorial-1198.html" \o "CSS Tutorial)

1. **Question 20. How To Implement Jwt Authentication In Express App ? Explain With Example?**

**Answer :**

* 1. Create a folder called ‘keys’ inside project folder.
  2. Install some dependencies as following:

  Npm install jsonwebtoken –save

* 1. Add the login router routes/index.js  
       
     router.post('/login, function(req, res) {  
        // find the user  
      User.findOne({  
        name: req.body.username  
      }, function(err, res) {  
        if (err) throw err;  
        if (!res) {  
          res.json({ success: false, message: Login failed.' });  
        } else if (res) {   
           // check if password matches  
          if (res.password != req.body.password) {  
           res.json({ success: false, message: Login  failed. Wrong password.' });  
          } else {  
            var token = jwt.sign(res, app.get('superSecret'), {  
              expiresInMinutes: 1600   
             });  
           // return the information including token as JSON  
            res.json({  
              success: true,  
             message: 'Valid token!',  
              token: token  
            });  
          }     
        }  });  
     });
  2. Use the token in application  
       
     jwt = require("express-jwt");  
            app.use(function(req, res, next) {  
            var token = req.body.token || req.query.token ||  req.headers['x-access-token'];   
       if (token) {  
            jwt.verify(token, app.get('superSecret'), function(err, decoded) {        
          if (err) {  
                return res.json({ success: false, message: 'Invalid token.' });      
         } else {  
            req.decoded = decoded;      
            next();  
          }  
        });  
      } else {  
        return res.status(403).send({   
             success: false,   
             message: 'No token given.'   
         });      
      }  
     });

[Javascript Objects Interview Questions](https://www.wisdomjobs.com/e-university/javascript-objects-interview-questions.html" \o "Javascript Objects Interview Questions)

1. **Question 21. How Should I Structure My Application?**

**Answer :**

There is no definitive answer to this question. The answer depends on the scale of your application and the team that is involved. To be as flexible as possible, Express makes no assumptions in terms of structure.

Routes and other application-specific logic can live in as many files as you wish, in any directory structure you prefer. View the following examples for inspiration:

* 1. Route listings
  2. Route map
  3. MVC style controllers

Also, there are third-party extensions for Express, which simplify some of these patterns:

* 1. Resourceful routing

1. **Question 22. How Do I Define Models?**

**Answer :**

Express has no notion of a database. This concept is left up to third-party Node modules, allowing you to interface with nearly any database.

[Javascript Objects Tutorial](https://www.wisdomjobs.com/e-university/javascript-objects-tutorial-1213.html" \o "Javascript Objects Tutorial)

1. **Question 23. How Can I Authenticate Users?**

**Answer :**

Authentication is another opinionated area that Express does not venture into. You may use any authentication scheme you wish.

[Javascript Advanced Interview Questions](https://www.wisdomjobs.com/e-university/javascript-advanced-interview-questions.html" \o "Javascript Advanced Interview Questions)

1. **Question 24. Which Template Engines Does Express Support?**

**Answer :**

Express supports any template engine that conforms with the (path, locals, callback) signature.

[MySQL Interview Questions](https://www.wisdomjobs.com/e-university/mysql-interview-questions.html" \o "MySQL Interview Questions)

1. **Question 25. How Do I Render Plain Html?**

**Answer :**

There’s no need to “render” HTML with the res.render() function. If you have a specific file, use the res.sendFile() function. If you are serving many assets from a directory, use the express.static() middleware function.

-------------------------------------------------------------------------------------------------------------------------------------------

**What is Express.js? What are core features of Express framework?**

**What is Express.js?**

Express.js is a light-weight node.js based web application framework. This JavaScript framework provides a number of flexible and useful feature in order to develop mobile as well as web application using NodeJS.

**ExpressJS Features:**

Following are some of the core features of Express framework −

* Set up middlewares in order to respond to HTTP/RESTful Requests.
* It is possible to defines a routing table in order to perform different HTTP operations.
* Dynamically renders HTML Pages based on passing arguments to templates.
* Provides all the feature provides by core Node.js.
* Express prepare a thin layer, therefore, the performance is adequate.
* Organize the web application into an MVC architecture.
* Manages everything from routes to rendering view and preforming HTTP request.

[Back to top](http://www.webdevelopmenthelp.net/2017/02/express-js-interview-questions.html#top)

**How to setup an Express.js App?**

We can follow the below step by step approach to set up an Application using ExpressJS Framework.

* Create a folder with the same name as Project name.
* Inside the folder create a file called ***package.json***.



|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25  26  27  28  29  30  31  32  33  34  35  36  37  38  39  40  41  42 | {    "name": "npm\_smart\_grocery",    "version": "1.0.0",    "description": "a sample smart grocery manager ",    "main": "index.js",    "": {      "ajv": "^4.9.0",      "async": "^1.4.2",      "body-parser": "^1.13.3",      "cloudant": "^1.4.0",      "dotenv": "^2.0.0",      "express": "^4.13.3",      "express-session": "^1.11.3",      "memory-cache": "^0.1.4",      "moment": "2.10.6",      "passport": "^0.3.2",      "path-exists": "^3.0.0",      "r-script": "0.0.3",      "rio": "^2.4.1",      "rox": "0.0.1-13",      "superagent": "^1.3.0",      "twitter": "^1.4.0",      "underscore": "^1.8.3",      "v8": "^0.1.0",      "winston": "^2.1.1",      "winston-daily-rotate-file": "^1.0.1"    },    "devDependencies": {},    "scripts": {      "test": "echo \"Error: no test specified\" && exit 1"    },    "repository": {      "type": "git",      "url": "git+<https://github.com/dahlia05/npm_smart_grocery.git>"    },    "author": "dahlia",    "license": "ISC",    "bugs": {      "url": "<https://github.com/dahlia05/npm_smart_grocery/issues>"    },    "homepage": "<https://github.com/dahlia05/npm_smart_grocery#readme>"  } |

* Open command prompt on the project folder and run following command.



|  |  |
| --- | --- |
| 1 | npm install |

This will install all the libraries defined in ***package.json*** inside **dependencies{}**and the libraries are installed in ***node\_modules*** folder.

* Create a file called ***server.js***.

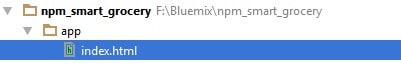


|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25  26  27  28  29  30  31  32  33  34  35  36  37  38  39  40  41  42  43  44  45  46  47  48  49  50  51  52  53  54 | "use strict";    var express = require('express');  var bodyParser = require('body-parser');  var session = require('express-session')      //var Project = require('./schema').Project;  var http = require('http');  var https = require('https');    /\* Initialize express app \*/  var app = express();    var host = process.env.APP\_HOST || 'localhost';  var port = process.env.APP\_PORT || '3000';  app.use(bodyParser.json());  app.use(session({     rolling: true,     saveUninitialized: true,     resave: true,     secret: config.SESSION\_SECRET,     cookie: {        maxAge: 36000000,        httpOnly: false     }  }));    app.use(express.static(\_\_dirname + '/app'));    var index = require('./routes/index');    app.use ('/’, index);    app.all('\*', function(req, res, next) {     res.set('Access-Control-Allow-Origin', '\*');     res.set('Access-Control-Allow-Methods', 'GET, POST, DELETE, PUT');     res.set('Access-Control-Allow-Headers', 'X-Requested-With, Content-Type');     if ('OPTIONS' == req.method) return res.send(200);     next();  });    /\* Define fallback route \*/  app.use(function(req, res, next) {      res.status(404).json({errorCode: 404, errorMsg: "route not found"});  });    /\* Define error handler \*/  app.use(logging.errorHandler);    /\* Start server \*/  app.listen(app.get('port'), function() {      logger.info('Express server listening on http://' + app.get('host') + ':' + app.get('port'));  }); |

* Create a folder called ‘routes’ inside the project folder.
* Create a file inside ‘routes’ folder called ***index.js***.



|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9 | var express = require('express');  var router = express.Router();    /\* GET home page. \*/  router.get('/', function(req, res, next) {    res.render('index', { title: 'Express' });  });    module.exports = router; |

* Create a folder called ‘app’ inside the project folder and create a file inside ‘app’ folder called ‘index.html’.
* Create a index.html file.



|  |  |
| --- | --- |
| 1 | <h1> Hello World!</h1> |

* Open command prompt on the project folder and run following command.



|  |  |
| --- | --- |
| 1 | node server.js |

The output would be



|  |  |
| --- | --- |
| 1 | 2017-02-01T21:31:15.889Z - info: Express server listening on http://localhost:3000 |

* Now open a browser with link [http://localhost:3000](http://localhost:3000/), the output would be:

[Back to top](http://www.webdevelopmenthelp.net/2017/02/express-js-interview-questions.html#top)

**How to config Angular 2.0 Frontend in Express.js Application?**

With the Angular 2.0 the MEAN stack has a huge changes. These are the following steps to create an application using MEAN stack.

* Install Node.js.
* Create an express application using express generator.
* Use the view engine as ejs, therefore, install ejs.



|  |  |
| --- | --- |
| 1 | npm install –save ejs |

* Set view engine in app.js.



|  |  |
| --- | --- |
| 1  2  3  4 | // view engine setup  app.set('views', path.join(\_\_dirname, 'views'));  app.set('view engine', 'ejs');  app.engine('html', require('ejs').renderFile); |

* Set the static folder.



|  |  |
| --- | --- |
| 1  2 | // Set Static Folder  app.use(express.static(path.join(\_\_dirname, 'public'))); |

* Create the config file inside public folder following the Getting started from angular.io.  
  ***package.json***



|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25  26  27  28  29  30  31  32  33  34  35  36  37 | {    "name": "todoapp",    "version": "1.0.0",    "scripts": {      "start": "tsc && concurrently \"tsc -w\" \"lite-server\" ",      "lite": "lite-server",      "postinstall": "typings install",      "tsc": "tsc",      "tsc:w": "tsc -w",      "typings": "typings"    },    "license": "ISC",    "dependencies": {      "@angular/common": "~2.0.1",      "@angular/compiler": "~2.0.1",      "@angular/core": "~2.0.1",      "@angular/forms": "~2.0.1",      "@angular/http": "~2.0.1",      "@angular/platform-browser": "~2.0.1",      "@angular/platform-browser-dynamic": "~2.0.1",      "@angular/router": "~3.0.1",      "@angular/upgrade": "~2.0.1",      "angular-in-memory-web-api": "~0.1.1",      "bootstrap": "^3.3.7",      "core-js": "^2.4.1",      "reflect-metadata": "^0.1.8",      "rxjs": "5.0.0-beta.12",      "systemjs": "0.19.39",      "zone.js": "^0.6.25"    },    "devDependencies": {      "concurrently": "^3.0.0",      "lite-server": "^2.2.2",      "typescript": "^2.0.3",      "typings":"^1.4.0"    }  } |

***tsconfig.json***



|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12 | {    "compilerOptions": {      "target": "es5",      "module": "commonjs",      "moduleResolution": "node",      "sourceMap": true,      "emitDecoratorMetadata": true,      "experimentalDecorators": true,      "removeComments": false,      "noImplicitAny": false    }  } |

***typings.json***



|  |  |
| --- | --- |
| 1  2  3  4  5  6  7 | {    "globalDependencies": {      "core-js": "registry:dt/core-js#0.0.0+20160725163759",      "jasmine": "registry:dt/jasmine#2.2.0+20160621224255",      "node": "registry:dt/node#6.0.0+20160909174046"    }  } |

***system.config.js***



|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25  26  27  28  29  30  31  32  33  34  35  36  37  38  39  40  41  42  43 | /\*\*  \* System configuration for Angular samples  \* Adjust as necessary for your application needs.  \*/  (function (global) {    System.config({      paths: {        // paths serve as alias        'npm:': 'node\_modules/'      },      // map tells the System loader where to look for things      map: {        // our app is within the app folder        app: 'app',        // angular bundles        '@angular/core': 'npm:@angular/core/bundles/core.umd.js',        '@angular/common': 'npm:@angular/common/bundles/common.umd.js',        '@angular/compiler': 'npm:@angular/compiler/bundles/compiler.umd.js',        '@angular/platform-browser': 'npm:@angular/platform-browser/bundles/platform-browser.umd.js',        '@angular/platform-browser-dynamic': 'npm:@angular/platform-browser-dynamic/bundles/platform-browser-dynamic.umd.js',        '@angular/http': 'npm:@angular/http/bundles/http.umd.js',        '@angular/router': 'npm:@angular/router/bundles/router.umd.js',        '@angular/forms': 'npm:@angular/forms/bundles/forms.umd.js',        // other libraries        'rxjs':                      'npm:rxjs',        'angular-in-memory-web-api': 'npm:angular-in-memory-web-api',      },      // packages tells the System loader how to load when no filename and/or no extension      packages: {        app: {          main: './main.js',          defaultExtension: 'js'        },        rxjs: {          defaultExtension: 'js'        },        'angular-in-memory-web-api': {          main: './index.js',          defaultExtension: 'js'        }      }    });  })(this); |

* Now run ‘***npm install***’ command to install all the necessary dependencies.
* Change the ‘views/index.html’ folder with the following code.



|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24 | <html>    <head>      <title>MyTaskList</title>      <meta charset="UTF-8">      <meta name="viewport" content="width=device-width, initial-scale=1">      <link rel="stylesheet" href="bower\_components/bootstrap/dist/css/bootstrap.css">      <link rel="stylesheet" href="styles.css">      <!-- 1. Load libraries -->       <!-- Polyfill(s) for older browsers -->      <script src="node\_modules/core-js/client/shim.min.js"></script>      <script src="node\_modules/zone.js/dist/zone.js"></script>      <script src="node\_modules/reflect-metadata/Reflect.js"></script>      <script src="node\_modules/systemjs/dist/system.src.js"></script>      <!-- 2. Configure SystemJS -->      <script src="systemjs.config.js"></script>      <script>        System.import('app').catch(function(err){ console.error(err); });      </script>    </head>    <!-- 3. Display the application -->    <body>      <my-app>Loading...</my-app>    </body>  </html> |

* Create an app.component.



|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11 | import { Component } from '@angular/core';  import {TaskService} from './services/task.service';    @Component({    moduleId: module.id,    selector: 'my-app',    templateUrl: 'app.component.html',    providers:[TaskService]  })    export class AppComponent { } |

* Add the template.

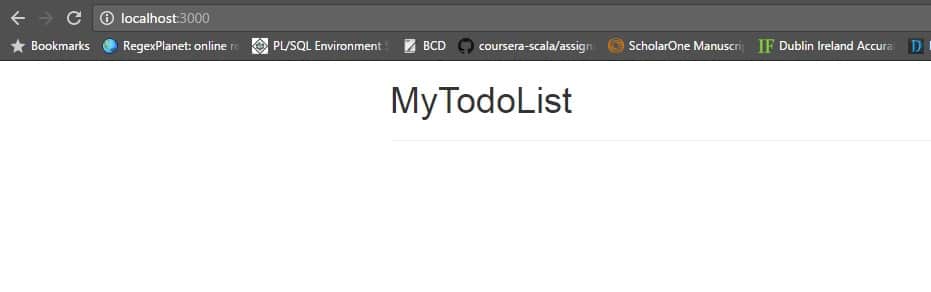


|  |  |
| --- | --- |
| 1  2  3  4  5 | <div class="container">      <h1>MyTodoList</h1>      <hr>    </div> |

* Create Angular App module in ‘public/app/app.module.ts’.



|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12 | import { NgModule }      from '@angular/core';  import { BrowserModule } from '@angular/platform-browser';  import {HttpModule} from '@angular/http';  import {FormsModule} from '@angular/forms';  import {AppComponent} from './app.component';    @NgModule({    imports:      [ BrowserModule, HttpModule, FormsModule ],    declarations: [AppComponent, TasksComponent],    bootstrap: [AppComponent]  })  export class AppModule { } |

* Now on the browser it would look like:
* Similarly add other angular module to the angular app.

[Back to top](http://www.webdevelopmenthelp.net/2017/02/express-js-interview-questions.html#top)

**Explain Routing in Express.js in details?**

* Create a file called routes.js  containing all the routes of the application.



|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25  26  27  28  29  30  31  32  33  34  35  36  37  38  39  40  41  42  43  44  45  46 | /\*\*  \* Contains all application routes.  \* @version 1.0  \*/  'use strict';    module.exports = {      '/member/register': {          post: {              controller: 'MemberController',              method: 'register',              public: true          }      },      '/member/login': {          post: {              controller: 'MemberController',              method: 'login',              public: true          }      },      '/member/logout': {          post: {              controller: 'MemberController',              method: 'logout',              public: true          }      },      '/member/profile/:profileId': {          post: {              controller: 'MemberController',              method: 'saveProfile',              public: false          },          get: {              controller: 'MemberController',              method: 'getProfile',              public: false          },          delete: {              controller: 'MemberController',              method: 'deleteProfile',              public: false          }      }  }; |

* Create a file called MemberController.js.



|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25  26  27  28  29  30  31  32  33  34  35  36  37  38  39  40  41  42  43  44  45  46  47  48  49  50  51  52  53  54  55  56  57  58  59  60  61  62  63  64  65  66  67  68  69  70  71  72  73  74  75  76  77  78  79  80  81  82  83  84  85  86  87  88  89  90  91  92  93  94  95  96  97  98  99  100  101  102  103  104  105  106  107  108  109  110  111  112  113  114  115  116  117  118  119  120  121  122  123  124  125  126  127  128  129  130  131  132  133  134  135  136  137  138  139  140  141  142  143  144  145  146  147  148  149  150  151  152  153  154  155  156  157  158  159  160  161  162  163  164  165 | /\*\*  \* This controller exposes REST actions for managing Games.  \*  \* @version 1.0  \*/  'use strict';    var async = require('async');  var MemberService = require('../services/MemberService');  var helper = require('../helpers/helper.js');  var DBService = require('../services/DBService');  var UserAlreadyExistsError = require('../helpers/errors').UserAlreadyExistsError;  /\*\*  \* register a member  \* @param {Object} req the request  \* @param {Object} res the response  \* @param {Function} next the next middleware  \*/  function register(req, res, next) {     var user =req.body;     async.waterfall([        function(cb) {           var query = { selector: { email: user.email }, limit: 1};           DBService.find(query , cb);        },        function(data, cb) {           if (typeof data.docs!== 'undefined' && data.docs.length > 0)              cb(new UserAlreadyExistsError());           else           {              MemberService.registerMember(user,cb);           }        },        function(result) {           res.json(helper.buildResponse(result));        }], function(err) {        if (err) {           return next(err);        }     });  }    /\*\*  \* Login.  \* @param {Object} req the request  \* @param {Object} res the response  \* @param {Function} next the next middleware  \*/  function login(req, res, next) {     var username = req.body.authInfo.username;     var password = req.body.authInfo.password;     async.waterfall([        function (cb) {           MemberService.loginMember(username,password, cb);        },        function (response, cb) {           var query = { selector: { email: username }, limit: 1};           DBService.find(query, cb);        },        function (data, cb) {           var user = {              "username": username,              "isLoggedIn": true           };           req.session.user = user;           res.json({              MemberInfo: user           });           cb();        }], function (err) {        if (err) {           req.session.destroy();           return next(err);        }     });  }    /\*\*  \* Logout.  \* @param {Object} req the request  \* @param {Object} res the response  \* @param {Function} next the next middleware  \*/  function logout(req, res, next) {     req.session.user = {};     res.end();  }    /\*\*  \* save profile  \*  \*/  function saveProfile(req, res, next) {     var profile = req.body;     var profileId = req.params.profileId;     var username = req.body.authInfo.username;     async.waterfall([        function (cb) {           MemberService.validateprofile(profile, cb);        }, function (profile, cb) {           MemberService.saveProfile(profile, profileId, cb);        },        function(result) {           res.json(helper.buildResponse({              profileId: profileId,              message: result           }));        }], function (err) {        if (err) {           req.session.destroy();           return next(err);        }     });  }    /\*\*  \* get profile  \*  \*/  function getProfile(req, res, next) {     var profileId = req.params.profileId;     var username = req.body.authInfo.username;     async.waterfall([        function (cb) {           MemberService.getProfile(profileId, cb);        },        function(result) {           res.json(helper.buildResponse(result));        }], function (err) {        if (err) {           req.session.destroy();           return next(err);        }     });  }    /\*\*  \* delete profile  \*  \*/  function deleteProfile(req, res, next) {     var profileId = req.params.profileId;     async.waterfall([        function (cb) {           MemberService.deleteProfile(profileId, cb);        },        function(result) {           res.json(helper.buildResponse(result));        }], function (err) {        if (err) {           req.session.destroy();           return next(err);        }     });  }    module.exports = {     login: login,     register:register,     logout: logout,     saveProfile:saveProfile,     getProfile:getProfile,     deleteProfile:deleteProfile    }; |

* Create a file called MemberService.js and DatabaseService.js would be created.
* In server.js add the following code:



|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25  26  27  28  29  30  31  32  33  34  35  36  37 | /\* Load all routes \*/  \_.each(require("./routes"), function (verbs, url) {      \_.each(verbs, function (def, verb) {       var method = require("./controllers/" + def.controller)[def.method];          if (!method) {              throw new Error(def.method + " is undefined");          }     var signature = def.controller + "#" + def.method;     var actions = [];     actions.push(function(req, res, next) {         req.signature = signature;         return next();     });     /\* If route is not public, check for user session first \*/     if (!def.public) {         actions.push(function(req, res, next) {        var user = req.session.user;        if (!user || !(user.isLoggedIn)) {            return next(new NotAuthenticatedError('You need to login first.'));        }        next();         });     }     actions.push(function(req, res, next) {         try {        logger.info('ENTER ' + signature);        method(req, res, next);        logger.info('EXIT ' + signature);         } catch(e) {        logging.logError(signature, e);        next(e);         }     });     app[verb](url, actions);      });  }); |

* Now any  of the route defined in routes.js can be accessed.

[Back to top](http://www.webdevelopmenthelp.net/2017/02/express-js-interview-questions.html#top)

**Explain Logging in Express.js? Give a practical example to demonstrate?**

With Express 4.0, the application can be generated using express-generator and it includes ***morgan***as the logger:

* Create express app using express generator.
* The middleware in app.js is already added.



|  |  |
| --- | --- |
| 1 | var logger = require('morgan'); |

* Create the local middleware.



|  |  |
| --- | --- |
| 1 | var logger = morgan('combined'); |

* Otherwise, If logging is need to be added to a log file.  
  Add fs to app.js



|  |  |
| --- | --- |
| 1 | var fs = require('fs') |

Add the file



|  |  |
| --- | --- |
| 1 | var log\_file = fs.createWriteStream(path.join(\_\_dirname, log.log'), {flags: 'a'}) |

Create the middleware



|  |  |
| --- | --- |
| 1 | Var logger = morgan('combined', {stream: log\_file}) |

* Make sure logging will be enabled only in development environment.



|  |  |
| --- | --- |
| 1 | app.use(logger('dev')); |

* Now if we run from the browser we can see that every request is being logged.



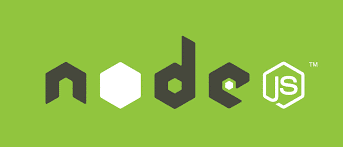
|  |  |
| --- | --- |
| 1  2  3  4  5  6  7 | GET /dsfsdf 500 387.461 ms - 1144  GET /stylesheets/style.css 304 3.383 ms - -  GET / 304 40.564 ms - -  GET /stylesheets/style.css 304 1.791 ms - -  GET /todos 200 1.397 ms - 51  GET /todos/new 304 62.912 ms - -  GET /stylesheets/style.css 304 0.397 ms - - |

[Back to top](http://www.webdevelopmenthelp.net/2017/02/express-js-interview-questions.html#top)

[**Learn to Build a Shopping Cart using NodeJS**](http://www.webdevelopmenthelp.net/2017/02/express-js-interview-questions.html#acc-2)

This mini [**NodeJS course**](https://click.linksynergy.com/deeplink?id=s8waqo9RNjU&mid=39197&murl=https%3A%2F%2Fwww.udemy.com%2Flearn-to-build-a-shopping-cart-using-nodejs%2F) will help you learn how to design your own eCommerce website using NodeJS, Express and Kraken. NodeJS is a brilliant open-source JavaScript runtime environment that allows developers to create powerful and dynamic server-side web applications.

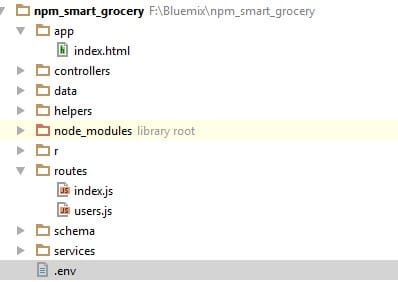
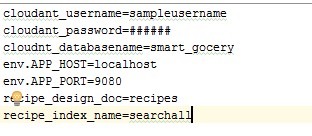
In this project-based course, you will build an entire Bookstore from scratch. You will be able to add, edit and delete books to the backend and customers will be able to browse, add and delete books from their cart as well as purchase it using Paypal.

[](https://click.linksynergy.com/deeplink?id=s8waqo9RNjU&mid=39197&murl=https://www.udemy.com/learn-to-build-a-shopping-cart-using-nodejs/)  
[**Take this Course Online Now**](https://click.linksynergy.com/deeplink?id=s8waqo9RNjU&mid=39197&murl=https%3A%2F%2Fwww.udemy.com%2Flearn-to-build-a-shopping-cart-using-nodejs%2F)

**How to config properties in Express Application?**

In an ExpressJS Application, we can config properties in following two ways:

**With Process.ENV?**

* Create a file with name ‘**.env**’ inside the project folder.
* Add all the properties in ‘**.env**’ file.
* In **server.js** any of the properties can be used as:



|  |  |
| --- | --- |
| 1  2  3 | var host = process.env.APP\_HOST  app.set('host', host);  logger.info('Express server listening on http://' + app.get('host')); |

**With RequireJs?**

* Create a file called  ‘**config.json**’ inside a folder called ‘config’ inside the project folder.
* Add config properties in **config.json**.



|  |  |
| --- | --- |
| 1  2  3  4 | {    "env":"development",    "apiurl":"<http://localhost:9080/api/v1/>"  } |

* Use require to access the config.json file.



|  |  |
| --- | --- |
| 1 | var config = require('./config/config.json'); |

[Back to top](http://www.webdevelopmenthelp.net/2017/02/express-js-interview-questions.html#top)

**How to allow CORS in ExpressJS? Explain with an example.**

In order to allow CORS in Express.js,  add the following code in server.js:



|  |  |
| --- | --- |
| 1  2  3  4  5  6  7 | app.all('\*', function(req, res, next) {     res.set('Access-Control-Allow-Origin', '\*');     res.set('Access-Control-Allow-Methods', 'GET, POST, DELETE, PUT');     res.set('Access-Control-Allow-Headers', 'X-Requested-With, Content-Type');     if ('OPTIONS' == req.method) return res.send(200);     next();  }); |

[Back to top](http://www.webdevelopmenthelp.net/2017/02/express-js-interview-questions.html#top)

**How to redirect 404 errors to a page in ExpressJS?**

In ***server.js*** add the following code to redirect 404 errors back to a page in our ExpressJS App:



|  |  |
| --- | --- |
| 1  2  3  4 | /\* Define fallback route \*/  app.use(function(req, res, next) {      res.status(404).json({errorCode: 404, errorMsg: "route not found"});  }); |

[Back to top](http://www.webdevelopmenthelp.net/2017/02/express-js-interview-questions.html#top)

**Explain Error Handling in Express.js using an example?**

From Express 4.0 Error handling is much easier. The steps are as following:

* Create an express.js application and as there is no built-in middleware like errorhandler in express 4.0, therefore, the middleware need to be either installed or need to create a custom one.

**Create a Middleware:**

* Create a middleware as following:



|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10 | // error handler  app.use(function(err, req, res, next) {    // set locals, only providing error in development    res.locals.message = err.message;    res.locals.error = req.app.get('env') === 'development' ? err : {};      // render the error page    res.status(err.status || 500);    res.render('error');  }); |

**Install Error Handler Middleware:**

* Install errorhandler.



|  |  |
| --- | --- |
| 1 | npm install errorhandler --save |

* Create a variable.



|  |  |
| --- | --- |
| 1 | var errorhandler = require('errorhandler') |

* Use the middleware as following:



|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13 | if (process.env.NODE\_ENV === 'development') {    // only use in development    app.use(errorhandler({log: errorNotification}))  }    function errorNotification(err, str, req) {    var title = 'Error in ' + req.method + ' ' + req.url      notifier.notify({      title: title,      message: str    })  } |

[Back to top](http://www.webdevelopmenthelp.net/2017/02/express-js-interview-questions.html#top)

**How to implement File uploading and downloading with Express?**

Below we have explained the process to upload as well as download a file with ExpressJS App.

**Upload File in Express.js:**

* Install formidable.



|  |  |
| --- | --- |
| 1 | npm install --save formidable |

* Add the following code in server.js in order to upload a file.



|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15 | var formidable = require('formidable');  app.post('/', function (req, res){      var form = new formidable.IncomingForm();        form.parse(req);        form.on('fileBegin', function (name, file){          file.path = \_\_dirname + '/uploads/' + file.name;      });        form.on('file', function (name, file){          console.log('Uploaded Successfully! ' + file.name);      });      res.sendFile(\_\_dirname + '/index.html');  }); |

* Update the index.html as following:



|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12 | <!DOCTYPE html>  <html>  <head>      <title>Simple File Upload Example</title>  </head>  <body>  <form action="/" enctype="multipart/form-data" method="post">      <input type="file" name="upload" multiple>      <input type="submit" value="Upload">  </form>  </body>  </html> |

* On Browser run ‘[http://localhost:3000’](http://localhost:3000%E2%80%99/).

**Download File in Express.js:**

* Add the following code in server.js.



|  |  |
| --- | --- |
| 1  2  3  4  5  6  7 | var router = express.Router();  // ...  router.get('/:id/download', function (req, res, next) {      var filePath = "/my/file/path/...";      var fileName = "samplefile.pdf";      res.download(filePath, fileName);  }); |

[Back to top](http://www.webdevelopmenthelp.net/2017/02/express-js-interview-questions.html#top)

**How to enable debugging in express app?**

In different Operating Systems, we have following commands:

On Linux the command would be as follows:



|  |  |
| --- | --- |
| 1 | $ DEBUG=express:\* node index.js |

On Windows the command would be:



|  |  |
| --- | --- |
| 1 | set DEBUG=express:\* & node index.js |

From Webstrome IDE



|  |  |
| --- | --- |
| 1 | C:\Program Files (x86)\JetBrains\WebStorm 2016.2.4\bin\runnerw.exe" "C:\Program Files\nodejs\node.exe" --debug-brk=61081 --expose\_debug\_as=v8debug E:\Development\nodejd\library\bin\www |

[Back to top](http://www.webdevelopmenthelp.net/2017/02/express-js-interview-questions.html#top)

**How to implement JWT authentication in Express app ? Explain with example.**

* Create a folder called ‘keys’ inside project folder.
* Install some dependencies as following:



|  |  |
| --- | --- |
| 1 | Npm install jsonwebtoken –save |

* Add the login router routes/index.js



|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25  26 | router.post('/login, function(req, res) {    // find the user    User.findOne({      name: req.body.username    }, function(err, res) {      if (err) throw err;      if (!res) {        res.json({ success: false, message: Login failed.' });      } else if (res) {          // check if password matches        if (res.password != req.body.password) {          res.json({ success: false, message: Login  failed. Wrong password.' });        } else {          var token = jwt.sign(res, app.get('superSecret'), {            expiresInMinutes: 1600          });          // return the information including token as JSON          res.json({            success: true,            message: 'Valid token!',            token: token          });        }     }  });  }); |

* Use the token in application



|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19 | jwt = require("express-jwt");  app.use(function(req, res, next) {    var token = req.body.token || req.query.token || req.headers['x-access-token'];    if (token) {      jwt.verify(token, app.get('superSecret'), function(err, decoded) {        if (err) {          return res.json({ success: false, message: 'Invalid token.' });        } else {          req.decoded = decoded;          next();        }      });    } else {      return res.status(403).send({          success: false,          message: 'No token given.'      });    }  }); |

**Q.1) What is Express Js?**

1. Express js is a free open-source, light-weight node js based web application framework.
2. It is designed for building (single-page, multi-page, and hybrid) web applications and APIs.
3. It has been developed by TJ Holowaychuk in 2010 and written in JavaScript.

**Express Js Features:**

Following are some of the core features of Express framework:

1. **Middlewares:**Set up middlewares in order to respond to HTTP/RESTful Requests.
2. **Routing:**It is possible to defines a routing table in order to perform different HTTP operations.
3. **Templates:**Dynamically renders HTML Pages based on passing arguments to templates.
4. **High Performance:**Express prepare a thin layer, therefore, the performance is adequate.
5. **Database Support:**Express supports RDBMS as well as NoSQL databases.
6. **MVC Support:**Organize the web application into an MVC architecture.
7. Manages everything from routes to rendering view and preforming HTTP request.

**Q.2) How can we allow CORS in Express JS? Explain with an Example?**

In order to allow CORS in Express.js, add the following code in app.js:

* code
* source

1. app.all('\*', function(req, res, next) {
2. res.set('Access-Control-Allow-Origin', '\*');
3. res.set('Access-Control-Allow-Methods', 'GET, POST, DELETE, PUT');
4. res.set('Access-Control-Allow-Headers', 'X-Requested-With, Content-Type');
5. if ('OPTIONS' == req.method) return res.send(200);
6. next();
7. });

 You may also like - [**Node Js Interview Questions Answers**](https://www.fullstacktutorials.com/interviews/node-js-interview-questions-29.html)

**Q.3) What is Scaffolding in Express JS?**

Scaffolding is creating the skeleton structure of application

There are 2 way to do this:

1. Express application generator
2. Yeoman

**1. Express application generator:**

Use express-generator to quickly create an application skeleton.

* code
* source

1. npm install express-generator -g
2. express myTestExpressApp

Th above command will create your project named - myTestExpressApp with following the files/folders in project.

1. **bin:**The **bin** folder have one file called **www** is the main configuration file of our app.
2. **public:**Public folder contains JavaScript, CSS, images etc.
3. **routes:** Routes folder contains routing files.
4. **views:** View folder contains the view files of the application.
5. **app.js:** The app.js file is the main file of the application.
6. **package.json:**package.json file is the manifest file. It contains all metadata of the project such as the packages used in the app (called dependencies) etc.

Install all the dependencies mentioned in the package.json file:

* code
* source

1. cd myTestExpressApp
2. npm install

**Q.4) How to enable debugging in express app?**

In different Operating Systems, we have following commands:

**On Linux:**

* code
* source

1. DEBUG=express:\*
2. node app.js

**On Windows:**

* code
* source

1. set DEBUG=express:\*
2. node app.js

**Q.5) Serving static files in Express?**

* code
* source

1. #Example:
2. app.use(express.static('public'))
3. app.use('/static', express.static(path.join(\_\_dirname, 'public')))

**Q.6) What is routing and how routing works in Express.js?**

Routing refers to determining how an application responds to a client request to a particular endpoint, which is a URI (or path) and a specific HTTP request method (GET, POST, and so on).

Each route can have one or more handler functions, which are executed when the route is matched.

**Route Syntax:**

* code
* source

1. app.METHOD(PATH, HANDLER)

**Where:**

* **app** is an instance of **express**.
* **METHOD** is an HTTP request method, in lowercase.
* **PATH** is a path on the server.
* **HANDLER** is the function executed when the route is matched.

**#Example:**

* code
* source

1. app.get('/', function (req, res) {
2. res.send('Express Js Interview Questions')
3. })

**Dynamic routing and how it works in Express.js?**

When someone pass parameters in URL i.e. Parametrized URL, this routing phenomenon is called dynamic routing.

* code
* source

1. var express = require('express'),
2. app = express();
3. app.get('/article/:id', function(req , res){
4. res.render('article' + req.params.id);
5. })

In above example: id is a parameters, which can be different for different calls.

**Q.7) What is middleware in Express Js?**

"A function that is invoked by the Express routing layer before the final request handler."

**Middleware functions can perform the following tasks:**

1. Execute any code.
2. Make changes to the request and the response objects.
3. End the request-response cycle.
4. Call the next middleware function in the stack.

*If the current middleware function does not end the request-response cycle, it must call next() to pass control to the next middleware function. Otherwise, the request will be left hanging.*

**An Express application can use the following types of middleware:**

1. Application-level middleware
2. Router-level middleware
3. Error-handling middleware
4. Built-in middleware
5. Third-party middleware

 You may also like - **[React Js Interview Questions](https://www.fullstacktutorials.com/interviews/react-js-interview-questions-answers-31.html)**

**1. Application-level middleware:**

This kind of middleware method is bind to the app Object using app.use() method.

* code
* source

1. *//This middleware will execute for each route.*
2. app.use(function (req, res, next) {
3. console.log('Current Time:', Date.now())
4. next()
5. })

**2. Router-level middleware:**

Router-level middleware works in the same way as application-level middleware, except it is bound to an instance of express.Router()

**3. Built-in middleware:**

Starting with version 4.x, Express no longer depends on **[Connect](https://github.com/senchalabs/connect" \t "_blank)**.

Express has the following built-in middleware functions:

* **express.static** serves static assets such as HTML files, images, and so on.
* **express.json** parses incoming requests with JSON payloads. **NOTE: Available with Express 4.16.0+**
* **express.urlencoded** parses incoming requests with URL-encoded payloads. **NOTE: Available with Express 4.16.0+**

**4. Third-party middleware:**

There are a number of third party middleware, such as body-parser cookie-parser, mongoose and so on.

To handle HTTP POST request in Express.js version 4 and above, you need to install middleware module called body-parser, body-parser extract the entire body portion of an incoming request stream and exposes it on req.body, The middleware was a part of Express.js earlier but now you have to install it separately.

These can be installed by using command:

* code
* source

1. >> npm install MODULE\_NAME

And they can be loaded using requires and used later.

#Example:

* code
* source

1. var bodyParser = require('body-parser');
2. app.use(bodyParser.json());
3. app.use(bodyParser.urlencoded({ extended: false }))

**Q.8) Database integration in Express Js?**

Adding the capability to connect databases to Express apps is just a matter of loading an appropriate Node.js driver for the database in your app

Express Js supports many RDBMS & NoSQL Ddatabase like

* MongoDB
* MySQL
* Oracle
* PostgreSQL
* SQL Server
* SQLite

#Example: Install MongoDB

* code
* source

1. >>npm install mongodb
2. var MongoClient = require('mongodb').MongoClient
3. MongoClient.connect('mongodb:*//localhost:27017/test\_db'*, function (err, db) {
4. if (err) throw err
5. db.collection('mammals').find().toArray(function (err, result) {
6. if (err) throw err
7. console.log(result)
8. })
9. })

**Q.9) Error handling in Express Js and How to redirect 404 errors to a page in Express Js?**

* code
* source

1. var express = require('express'),
2. app = express();
3. app.use(function (err, req, res, next) {
4. console.error(err.stack)
5. res.status(500).send('Something went wrong, Express Js Interview Questions')
6. })

* code
* source

1. *//To redirect 404 errors*
2. var express = require('express'),
3. app = express();
4. app.use(function(req, res, next) {
5. res.status(404).json({
6. errorCode: 404,
7. errorMsg: "route not found"
8. });
9. });

**Q.10) How to implement JWT authentication in Express app ? Explain with an Example?**

* Create a folder called 'keys' inside project folder.
* Install some dependencies as following: npm install jsonwebtoken –-save
* Add the login router routes/index.js
  + code
  + source
  + router.post('/login, function(req, res) {
  + *// find the user*
  + User.findOne({
  + name: req.body.username
  + }, function(err, res) {
  + if (err) throw err;
  + if (!res) {
  + res.json({ success: false, message: Login failed.' });
  + } else if (res) {
  + *// check if password matches*
  + if (res.password != req.body.password) {
  + res.json({ success: false, message: Login failed. Wrong password.' });
  + } else {
  + var token = jwt.sign(res, app.get('superSecret'), {
  + expiresInMinutes: 1600
  + });
  + *// return the information including token as JSON*
  + res.json({
  + success: true,
  + message: 'Valid token!',
  + token: token
  + });
  + }
  + } });
  + });
* Use the token in application
  + code
  + source
  + jwt = require("express-jwt");
  + app.use(function(req, res, next) {
  + var token = req.body.token || req.query.token || req.headers['x-access-token'];
  + if (token) {
  + jwt.verify(token, app.get('superSecret'), function(err, decoded) {
  + if (err) {
  + return res.json({ success: false, message: 'Invalid token.' });
  + } else {
  + req.decoded = decoded;
  + next();
  + }
  + });
  + } else {
  + return res.status(403).send({
  + success: false,
  + message: 'No token given.'
  + });
  + }
  + });

**Q1. What is Express.js?**  
Express.js is a light-weight node.js based web application framework. This JavaScript framework provides a number of flexible and useful feature in order to develop mobile as well as web application using NodeJS.

**Q2. How to install express js in node?**  
use following command to install express js.  
npm install express

**Q3. What type of web application can built using Express JS?**  
you can build single-page, multi-page, and hybrid web applications.

**Q4. Why to use Express?**  
You should use Express because provides following features:  
1.Parses the arguments and headers  
2.Supports Routing  
3.Supports multiple view engines like Jade, EJS, JSHtml etc.  
4.Handle Configurations  
5.Supports Sessions  
6.Supports Content Negotiation  
7.Supports Error Handling  
8.Supports Multiple Databases  
o RDBMS – MySQL, MS SQL etc.  
o NoSQL – MongoDB, Firebase, Redis etc.

**Q5 List out the ExpressJS Features**  
Following are some of the core features of Express framework −

* Set up middlewares in order to respond to HTTP/RESTful Requests.
* It is possible to defines a routing table in order to perform different HTTP operations.
* Dynamically renders HTML Pages based on passing arguments to templates.
* Provides all the feature provides by core Node.js.
* Express prepare a thin layer, therefore, the performance is adequate.
* Organize the web application into an MVC architecture.
* Manages everything from routes to rendering view and preforming HTTP request.

**Q6. How to create an Http server using Express?**  
Express is best for developing web application using Node.js. You can create an http server using express as given below:  
var express = require(“express” );  
var app = express();  
app.get( “/”, function (req, res) {  
res.write(“Hello, Express”);  
res.end();  
});  
var port = process.env.port || 1305;  
app.listen(port);  
console.log(“Server is running at [http://localhost:”](http://localhost:%E2%80%9D/) + port);

**Q7. How To Install Expressjs?**  
Assuming you’ve already installed Node.js, create a directory to hold your application, and make that your working directory.  
$ mkdir myapp  
$ cd myapp  
Use the npm init command to create a package.json file for your application. For more information on how package.json works, see Specifics of npm’s package.json handling.

$ npm init  
This command prompts you for a number of things, such as the name and version of your application. For now, you can simply hit RETURN to accept the defaults for most of them, with the following exception:

entry point: (index.js)  
Enter app.js, or whatever you want the name of the main file to be. If you want it to be index.js, hit RETURN to accept the suggested default file name.

Now install Express in the myapp directory and save it in the dependencies list. For example:  
$ npm install express –save  
To install Express temporarily and not add it to the dependencies list, omit the –save option:  
$ npm install express

**Q8. What are different methods in REST API?**  
GET : Used to read.  
POST: Used to update.  
PUT: Used to create.  
DELETE: Used to delete

**Q9. How to use express js in node?**  
use require to include express module.  
var app = require(‘express’)();

**Q10. How to use handle get request in express Js?**  
/\*Include require module\*/  
var app = require(‘express’)();  
var http = require(‘http’).Server(app);  
app.get(‘/’, function (req, res) {  
console.log(“Got a GET request for the homepage”);  
//Shown in console res.send(‘This is GET Method for Homepage’); //  
Display as response  
})  
/\*Start listing 8080 port\*/  
http.listen(‘8080’, function() {  
console.log(‘listening on \*:8080’);  
});

**Q11. What Function Arguments Are Available To Express.js Route Handlers?**  
The arguments available to an Express.js route handler function are:

* req – the request object
* res – the response object
* next (optional) – a function to pass control to one of the subsequent route handlers

The third argument may be omitted, but is useful in cases where you have a chain of handlers and you would like to pass control to one of the subsequent route handlers, and skip the current one.

**Q12.What is template engine?**  
A template engine allows us to create and render an HTML template with minimal code. At runtime, a template engine executes expressions and replaces variables with their actual values in a template file. In this way, it transforms the template into an HTML file and sent to it the client.  
 **Q13. What template engines you can use with express?**  
The popular template engines which you can use with Express are Pug, Handlebars, Mustache, and EJS. The Express application generator uses Pug as its default template engine.

**Q14. How to download a file?**  
app.get(‘/download’, function(req, res){  
var file = \_\_dirname + ‘/download-folder/file.txt’;  
res.download(file);  
});

**Q15. How to get variables in Express.js in GET Method?**  
var express = require(‘express’);  
var app = express();  
app.get(‘/’, function(req, res){  
/\* req have all the values \*\*/  
res.send(‘id: ‘ + req.query.id);  
});  
app.listen(3000);

**Q16. How to get POST a query in Express.js?**  
var bodyParser = require(‘body-parser’) app.use( bodyParser.json() );  
// to support JSON-encoded  
app.use(bodyParser.urlencoded({  
// to support URL-encoded  
extended: true  
}));

**Q17. What is the parameter “next” used for in Express?**  
app.get(‘/userdetails/:id?’, function(req, res, next){ });  
**req** and **res** which represent the request and response objects  
**next**It passes control to the next matching route.

**Q18. What is body-parser**  
Parse incoming request bodies in a middleware before your handlers, available under the req.body property.  
To handle HTTP POST request in Express.js version 4 and above, you need to install middleware module called body-parser.  
body-parser extract the entire body portion of an incoming request stream and exposes it on req.body.  
Installation  
$ npm install body-parser  
API  
var bodyParser = require(‘body-parser’)  
Example  
  
var express = require(‘express’)  
var bodyParser = require(‘body-parser’)  
var app = express()  
// parse application/x-www-form-urlencoded  
app.use(bodyParser.urlencoded({ extended: false }))  
// parse application/json  
app.use(bodyParser.json())  
app.use(function (req, res) {  
res.setHeader(‘Content-Type’, ‘text/plain’)  
res.write(‘you posted:\n’)  
res.end(JSON.stringify(req.body, null, 2))  
})

**Question: How to install express js in node?**  
use following command to install express js. 

npm install express

**Question: How to use express js in node?**  
use require to include express module. 

var app = require('express')();

**Question: How to use handle get request in express Js?**

/\*Include require module\*/

var app = require('express')();

var http = require('http').Server(app);

app.get('/', function (req, res) {

console.log("Got a GET request for the homepage"); //Shown in console

res.send('This is GET Method for Homepage'); //Display as response

})

/\*Start listing 8080 port\*/

http.listen('8080', function() {

console.log('listening on \*:8080');

});

**Question: How to use handle post request in express Js?**

/\*Include require module\*/

var app = require('express')();

var http = require('http').Server(app);

app.post('/user\_list', function (req, res) {

console.log("Got a POST request for the user\_list URL"); //Shown in console

res.send('This is POST Method for user\_list URL');//Display as response

})

/\*Start listing 8080 port\*/

http.listen('8080', function() {

console.log('listening on \*:8080');

});

**Question: How to use handle GET/POST request for same URL expressJs?**

/\*Include require module\*/

var app = require('express')();

var http = require('http').Server(app);

//This is POST Request

app.post('/add\_user', function (req, res) {

console.log("Got a POST request for the add\_user URL"); //Shown in console

res.send('This is POST Method for add\_user URL');//Display as response

})

//This is Get Request

app.get('/user\_list', function (req, res) {

console.log("Got a GET request for the user\_list URL"); //Shown in console

res.send('This is GET Method for user\_list URL');//Display as response

})

/\*Start listing 8080 port\*/

http.listen('8080', function() {

console.log('listening on \*:8080');

});

**Question: How to use handle GET request for all URL start with ab\* ?**

app.get('/ab\*', function(req, res) {

console.log("Got a GET request for /ab\*");

res.send('Page Regex Match');

})

**Question: How to send Server call in Node?**  
Install the "request" module with following command. 

npm install request

Include the request module in page, and send the request to server. 

request = require('request');

request( "http://exmaple.com:8081:/users/add/?n=test&p=2277585&email=test@gmail.com", function(err, res, body) {

console.log(res);

console.log(body);

});

**Question: How to post data and get in express Js?**  
HTML Code

<form action="http://example.com:8081/register" method="GET">

Name: <input name="name" type="text" /> <br />

Email: <input name="email" type="text" />

Phone: <input name="phone" type="text" />

<input type="submit" value="Submit" />

</form>

Express JS Code

app.get('/register', function (req, res) {

response = {

name:req.query.name,

email:req.query.email

phone:req.query.phone

};

console.log(response);

res.end(JSON.stringify(response));

})

**Question: How to get cookie in Express js?**  
Install the "cookie-parser" module with following command. 

npm install cookie-parser

Include the cookie-parser, in node.

app.get('/', function(req, res) {

console.log("Cookies: ", req.cookies)

})

**Question: How Static Files works in Express js?**  
Static files are images/videos/css/js etc.  
For this, first you need to set the folder path using "express.static" method. See Example

var express = require('express');

var app = express();

app.use(express.static('public'));

app.get('/', function (req, res) {

res.send('Hello ');

});

var server = app.listen(8081, function () {

var host = server.address().address

var port = server.address().port

})

**Question: How to upload images in Express js?**  
HTML Code 

<form action="http://example.com:8081/file\_upload" enctype="multipart/form-data" method="POST">

<input name="file" size="50" type="file" />

<input type="submit" value="Submit" />

</form>

ExpressJS Code

var fs = require("fs");

var express = require('express');

var app = express();

var fs = require("fs");

app.post('/file\_upload', function (req, res) {

console.log(req.files.file.name);

var destinationFile = \_\_dirname + "/images/" + req.files.file.name;

fs.readFile( req.files.file.path, function (err, data) {

fs.writeFile(destinationFile, data, function (err) {

if( err ){

console.log( err );

}else{

response = {

message:'File uploaded successfully in '+destinationFile,

filename:req.files.file.name

};

}

//console.log( response );

res.end( JSON.stringify( response ) );

});

});

})

**Question: What are different methods in REST API?**

1. GET : Used to read.
2. POST: Used to update.
3. PUT: Used to create.
4. DELETE: Used to delete

**Question: How to setup the images and access them?**  
Add Following code in my server.js (OR main file). 

app.use(express.static('public'))

Now, you can add images in public folder and can access like below:

http://localhost:3000/1.jpg

http://localhost:3000/2.jpg

http://localhost:3000/3.jpg

**Question: How to do increment the value by 1?**

User.update({id: 100}, {$inc: { views: 1 }}).limit(1).exec();

**Question: How to do decrement the value by 1?**

User.update({id: 100}, {$inc: { views: -1 }}).limit(1).exec();

**1) What is middleware functions in Express.js?**

Middleware is a functions that performs some preparatory or commonly used action before passing request to main handler. For example you need to check user session on every request, so you write middleware functions, that check it, so you don't duplicate this check in all controllers.

**import** \* **as** express **from** 'express';

**const** app = express();

app.use(**function**(req, res, next) {

**const** token = req.headers['token'] || req.cookies.token;

**let** result = tokenService.verifyToken(token);

**if** (result.error) {

**return** NotAuthorized(req, res);

}

req.user = result;

**return** next(null, req, res);

});

**2) How to make a redirect to specific URL?**

To redirect to specific URL use response method redirect([status,] path). It accepts one two parameters: URL,redirect status code or just URL.

res.redirect(301, '[https://domain.com](https://domain.com/)');

res.redirect('[https://domain.com](https://domain.com/)');

**3) How to get query params?**

You can get query params in req.params object.

router.**get**('/api/user/:userId', **async** **function**(req, res, next) {

**const** userId = req.**params**.userId;

});

**4) What is template engines and how to use them?**

A template engine allows you to use static template files in your application to render dynamic HTML responses. To use template engine use app.set() method.

app.**set**('view engine', 'hbs')

**5) How to run Express.js server in production mode?**

To run Express server in production mode, you should set NODE\_ENV variable to 'production'. By default server works in development mode.

$ NODE\_ENV**=**production node bin/www

**6) How to automatically redirect HTTP traffic to HTTPS?**

There is couple of way to do it. First is to utilize res.redirect method, second to use Nginx or HAProxy as reverse proxy

**const** http = express.createServer();

http.get('\*', **function**(req, res) {

res.redirect('https://' + req.headers.host + req.url);

});

http.listen(80);

For Ngnix:

**location** / {

**proxy\_pass** http: //localhost:8080;

}

**7) How to run Express.js app in cluster mode?**

It's well know fact, that Node.js is single threaded, so your can't fully utilize multi-core system capacities running in single mode. To run multiple instance of Express.js built in 'cluster' module should be used.

**const** cluster = require('cluster');

**if** (cluster.isMaster) {

**const** cpuCount = require('os').cpus().length;

**for** (**var** i = 0; i < cpuCount; i += 1) {

cluster.fork();

}

} **else** {

**const** express = require('express');

**const** app = express();

app.get('/', **function**(req, res) {

res.end('Rendered in cluster mode.!');

});

app.listen(8080);

}

**8) How to handle file upload?**

Express doesn't provide file upload api out the box, but you can use one of various middlewares, which are availble at NPM. In following example we use a 'multer' package.

**import** \* **as** express **from** 'express'

**const** router = express.Router();

**import** \* **as** multer **from** "multer"

**const** fs = require('fs');

**const** appRoot = require('app-root-path').path;

**const** upload = multer({

dest: appRoot + '/uploads'

});

router.post('/api/v1/upload/image', upload.single('image'), **function**(req: any, res, next) {

**const** sanitize = require("sanitize-filename");

**if** (!req.file) {

**return** res.json({

error: 'No image provided'

});

}

**const** newFileName = `${req.file.path}\_${sanitize(req.file.originalname)}`;

fs.rename(req.file.path, newFileName, **function**(err) {

**if** (err) console.log('ERROR: ' + err);

});

res.send({

status: "OK"

});

});

**export** {

router

};

**9) ExpressJS how to structure an application?**

There is no official recommendation about how to structure Express.js application, so creating architecture rests on the shoulders of developer. There is some recommendation about directory structure.

├── app.js // file where express server started

├── config // Configuration files

├── models // Data models **for** Sequlize or Mongoose

├── **public** // Directory **for** static assets

├── routes // The routes in you application, with controller functions

├── services // Any other service utils your application using

└── views // Template engine views **(**HBS,JADE,EJS**)**

**10) How to restart Node.js application when uncaught exception happens?**

When application works in production, uptime should be as big as possible, so there is a need of service that will restart server after critical error. There is several command line utils, which can do this: forever,pm2,nodedemon. The main difference is that you start application through this util,you don't run it with 'node' command directly. Also this packages save logs of application, provides information about memory usage and current uptime.

**pm2** **start** **app**.js

**11) Does Express.js supports basic auth?**

No, express doesn't support it out of the box, but there is library express-basic-auth, which helps to add this feature to app.

**12) What res.end() function do?**

res.end function ends response process. It can be used to quickly send response with no data or when you to signal that you write last chunk of buffered data.

app.**use**(**function**(req, res) {

res.status(404).end();

})

**13) How to send JSON response?**

The easiest way to send JSON response is to use res.json(data) method. It will automatically add correct content-type to response headers and transforms passed object to JSON. However you could it on your own using res.send() and setting headers.

router.get('/api/image-dimensions', upload.single('image'), **function**(req, res, next) {

**const** json = getImageDimension();

res.header('Content-Type', 'application/json');

res.send(json);

})

**14) What res.type(type) function do?**

Function res.type(type) change the Content-Type HTTP header to the MIME type as determined by mime.lookup() for the specified type.

res.type('html');*// => 'text/html'*

**15) Describe main differences between and Express.js and Sails.js?**

* Sails.js is more powerful, it have own ORM, can auto-generate REST API's, built'n WebSocket support, defined application structure. Express.js can't do nothing similar without additional libraries and middlewares.
* Express.js is more flexible than Sails.js, you can build your app like a constructor. Sails.js severely limits developer ability to change application structure.
* Express.js is easier to learn, which is great for beginner programmers.
* Sails.js has powerful CLI utility

**16) What is 'express-generator'?**

Express generator is util to generate express.js apps. Before project generation you can configure template engine,stylesheet engine, .gitignore file.

$ npm install -g express-generator

$ express --view**=**hbs /tmp/foo **&&** cd /tmp/foo

**17) What is router?**

A router object is an isolated instance of middleware and routes. Router is a basic class to implement REST api and Website pages. router.METHOD(path, [callback, ...] callback) method is used to add route.

**18) Could route handler function be async?**

Yes, it's is possible to utilize async/await in express.js controller functions.

**19) How to send file using Node.js streams ?**

Node.js streams allows read and write buffered data. For example it's very useful when file doesn't feet in ram, so you don't reed full file, just process it in chunks.

router.get('/api/v1/image-proxy/:name', **function**(req, res) {

**return** require('request').get('some url' + req.params.name).pipe(res);

})

**20) Why is important to distinguish operational and programmer errors ?**

Operational errors are not bugs, but problems with the system, like down database or full hard drive, programmer error is a bug which leads to incorrect application behaviour. It's good practise to be ready to any operational error and handle gracefully without crashing.

**Question: What is socket.io module?**  
Socket.IO is a JavaScript library for realtime web applications.  
It enables realtime, bi-directional communication between web clients and servers.  
It has two parts  
**a**) client-side library that runs in the browser.  
**b**) server side that run in browser.   
  
  
  
**Question: Give example of real time application?**

1. Instant messengers like facebook notification/message.
2. Push Notifications like when someone tags you in a picture on facebook, you receive an instant notification.
3. Collaboration applications like multiple user editing same file.
4. Online gaming

**Question: Why we use socket.io?**

1. bi-directional communication channel between a client and a server.
2. When an event occur, socket get notify and let to know the single client OR all clients.
3. It is trust application and used by Microsoft Office, Yammer, Zendesk, Trello, and numerous other organizations.

**Question: How to install socket.io in Web and Server?**  
**In Browser (client)**

<script src="https://cdnjs.cloudflare.com/ajax/libs/socket.io/2.0.3/socket.io.js"></script>

**In NodeJS**

npm install socket.io

var io = require('socket.io')

**Question: How get to know, if user is connect/disconnect?**  
In Server(Node) 

var app = require('express')();

var http = require('http').Server(app);

var io = require('socket.io')(http);

app.get('/', function(req, res){

res.sendfile('index.html');

});

//Whenever someone connects (execute when come in Web page)

io.on('connection', function(socket){

console.log('A user connected');

//Whenever someone disconnects (execute when Leave the web page)

socket.on('disconnect', function () {

console.log('A user disconnected');

});

});

http.listen(3000, function(){

console.log('listening on \*:3000');

});

In Browser

var socket = io.connect('http://127.0.0.1:3000');

**Question: How to get to know, when new user comes in browse?**  
  
OR  
**Question: How to send message from Client(Browser) to Server(Node)?**  
In Server

socket.on('adduser', function(name, email, uid) {

//name, email, and uid of user

console.log('New user comes in web page');

});

In Browser>

socket.emit('adduser', 'Robin Sharma','robinsharma@no-spam.ws',2555);

**Question: How to send message from Server(Node) to Client(Browser)?**  
In Server

socket.emit('newRequestCome', 'name','myself send request'); //Send message to socket (Same window)

socket.broadcast.to(roomName).emit( 'newRequestCome','name','message'); //Send message to all socket (Other windows in same Room)

In Browser

socket.on('newRequestCome', function(name, message) {

});

**Question: How to join the room and send message?**  
Join the room

io.on('connection', function(socket){

socket.join('someNo12');

});

**Question: How to send message the room?**  
Send message to the room

io.in('someNo12').emit('some event');

//io.to('someNo12').emit('some event');

**Difference Between WebSocket vs Socket.io**

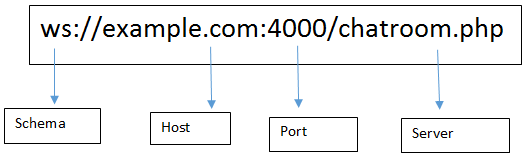
**WebSocket :**

It is the communication Protocol which provides bidirectional communication between the Client and the Server over a TCP connection, WebSocket remains open all the time so they allow the real-time data transfer. When clients trigger the request to the Server it does not close the connection on receiving the response, it rather persists and waits for Client or server to terminate the request.

**Key features of WebSocket :**

* WebSocket helps in [real-time communication](https://www.educba.com/real-time-analytics/) between the Client and the web server.
* This protocol helps in transforming to cross-platform in a real time world between the server and the client.
* This also enables the business around the world for real-time [web application to enhance](https://www.educba.com/how-to-build-web-applications-using-mongodb/) and to increase the feasibility.
* The major advantage it stands over an HTTP connection that it provides full duplex communication.

**WebSocket protocol schema :**



**Why do we need WebSocket:**

* It provides the full duplex communication which helps in persisting the connection established between the Client and the Web Server.
* It also lives up to the standards and provides the accuracy and efficiency stream events to and from with negligible latency.
* WebSocket removes the overhead and reduce complexity.
* It makes real-time communication effortless and efficient.

**Socket.IO:**

It is a library which enables real-time and full duplex communication between the Client and the Web servers. It uses the [WebSocket protocol](https://www.educba.com/websocket-vs-rest/) to provide the interface. Generally, it is divided into two parts, both WebSocket vs Socket.io are event-driven libraries

* Client Side: it is the library that runs inside the browser
* Server Side: It is the library for Node.js

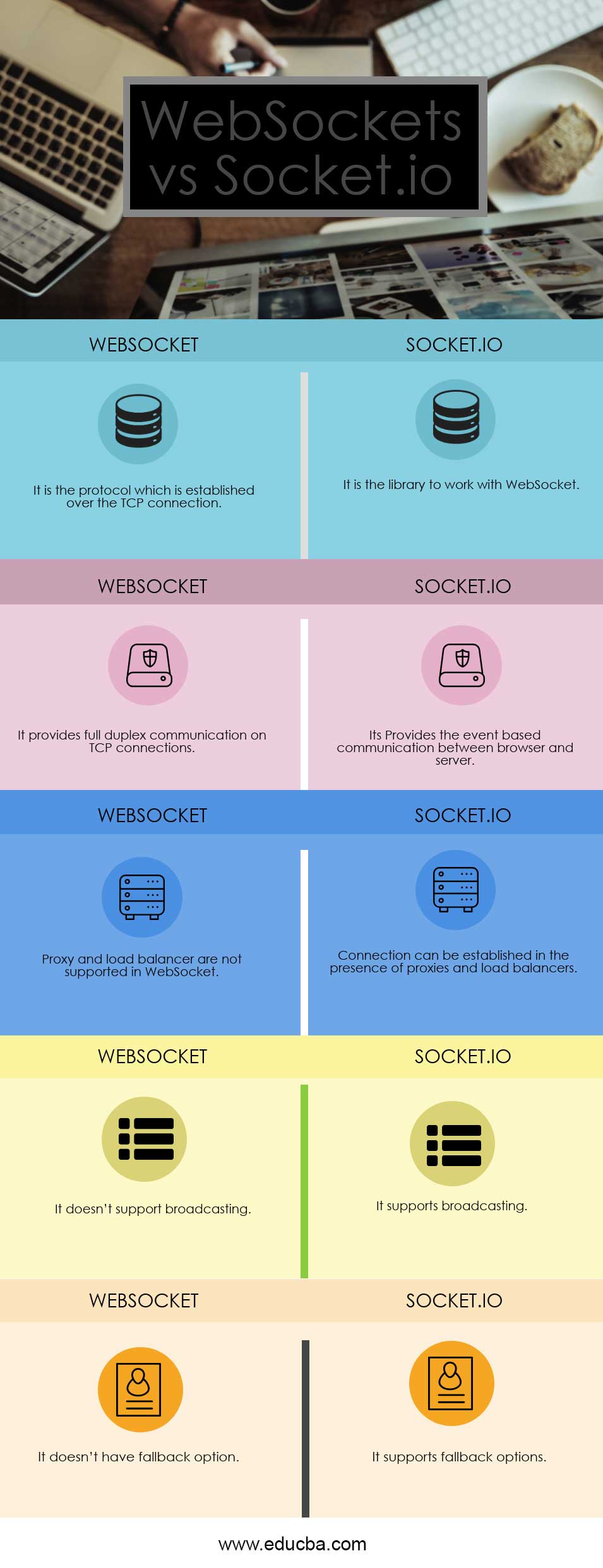
**Key features of Socket.IO:**

* It helps in broadcasting to multiple sockets at a time and handles the connection transparently.
* It works on all platform, server or device ensuring the equality, reliability, and speed.
* It automatically upgrades the requirement to WebSocket if needed.
* It is a custom real-time transport protocol implementation on top of other protocols.
* It requires both libraries to be used Client side as well as a server-side library.
* IO works on work-based events. there are some reserved events which can be accessed using the Socket on server side like Connect, message, Disconnect, Ping and Reconnect.
* There are some Client based reserved events like Connect, connect- error, connect-timeout and Reconnect etc.

**Why do we need Socket.IO:**

* I handle all the degradation of your technical alternatives to get full duplex communication in real time.
* It also handles the various support level and the inconsistencies from the browser.
* It also gives the additional feature room support for basic publish infrastructure and thinks like automatic reconnect.
* Currently, AFAIK is the most used one and easier to help with vanilla web sockets.

**Head To Head Comparison Between WebSocket vs Socket.io (Infographics)**

Below is the top 5 comparison of WebSocket vs Socket.io:

**Key Differences between WebSocket vs socket.io**

Both WebSocket vs Socket.io are popular choices in the market; let us discuss some of the major Difference Between WebSocket vs Socket.io:

* It provides the Connection over TCP while Socket.io is a library to abstract the WebSocket connections.
* WebSocket doesn’t have fallback options while Socket.io supports fallback.
* WebSocket is technology while Socket.io is a [library for WebSockets](https://www.educba.com/websockets-vs-webrtc/).

**WebSocket vs Socket.io Comparison Table**

Below is the topmost comparison between WebSocket vs Socket.io:

|  |  |  |
| --- | --- | --- |
| **S. No.** | **WebSocket** | **Socket.io** |
| 1 | It is the protocol which is established over the TCP connection | It is the library to work with WebSocket |
| 2 | It provides full duplex communication on TCP connections. | Provides the event-based communication between browser and server. |
| 3 | Proxy and load balancer is not supported in WebSocket. | A connection can be established in the presence of proxies and load balancers. |
| 4. | It doesn’t support broadcasting. | It supports broadcasting. |
| 5. | It doesn’t have a fallback option. | It supports fallback options. |

**Conclusion – WebSocket vs Socket.io**

Top Socket Io interview questions and answers for freshers and experienced

What is Socket Io ?

**Answer :**Socket.IO aims to make realtime apps possible in every browser and mobile device, blurring the differences between the different transport mechanisms. It supports multiple transports, such as WebSockets, Flash sockets, long polling and more, automatically falling back when a transport fails

<https://www.tutorialspoint.com/socket.io/>

Top Answers to MongoDB Interview Questions

**1. Compare MongoDB and Cassandra**

|  |  |  |
| --- | --- | --- |
| **Criteria** | **MongoDB** | **Cassandra** |
| Data Model | Document | Big Table like |
| Database scalability | Read | Write |
| Querying of data | Multi-indexed | Using Key or Scan |

To learn more about the comparison of MongoDB & Cassandra check [Cassandra Versus MongoDB.](https://intellipaat.com/blog/cassandra-vs-mongodb-differences/)

**2. What makes MongoDB the best?**

MongoDB is considered to be best NoSQL database because of :Document-oriented (DO)  
High performance (HP)  
High availability (HA)  
Easy scalability  
Rich query language

**3. How to do transactions/locking in MongoDB?**

MongoDB does not use conventional locking with reduction, as it is planned to be light, high-speed and knowable in its presentation. It can be considered as parallel to the MySQL MyISAM auto entrust sculpt. With simplest business sustain, performance is enhanced, particularly in a structure with numerous servers.

Check out this video on MongoDB Tutorial for Beginners

[](https://intellipaat.com/interview-question/mongodb-interview-questions/#login-form)

Learn for free ! Subscribe to our youtube Channel.

[](https://www.youtube.com/user/intellipaaat?sub_confirmation=1)

**4. When and to what extent does Data get extended to Multi-slice?**

The MongoDB scrap stands on a collection. So an album of all substances is kept in a lump or mass. Only when there is an additional time slot, there will be more than a few slice data achievement choices, but when there is more than 1 lump, data gets extended to a lot of slices and it can be extended to 64 MB.

**5. Judge against MongoDB with Couchbase and CouchbaseDB?**

Although Mongo DB with Couchbase and Couchbase DB are common in many ways, but still they are different in the case of necessities for execution of the model, crossing points, storage, duplications, etc.

**6. When do we use Namespace in MongoDB?**

During sequencing of the names of the database and collection name Namespace is used.

**7. If you remove an object attribute, is it deleted from the database?**

Yes, it is deleted. So better eliminate the attribute and then save the object again.

**8. How can we move the old file in the moveChunk directory?**

Once the functions are done, the old files are converted to backup files and moved to the moveChunk directory at the time of balancing the slices.

**9. Explain the situation when an index does not fit into RAM?**

When an index is too huge to fit into RAM, then MongoDB reads the index, which is faster than reading RAM because the indexes easily fit into RAM if the server has got RAM for indexes along with the remaining set.

[](https://intellipaat.com/mongodb-training/)

**10. How does MongoDB provide consistency?**

MongoDB uses the reader-writer locks, allowing simultaneous readers to access any supply like a database or any collection. But always offers private access to singles writes.

**11. Why is MongoDB not chosen for a 32-bit system?**

Mongo DB is not considered as a 32-bit system because for running the 32-bit MongoDB, with the server, information and indexes require 2 GB. So only it is not used in 32-bit devices.

**12. How does Journaling work in MongoDB?**

Write operations are saved in the memory while journaling is going on. The on-disk journal files are really dependable for the reason that the journal writes are habitual. Inside dbPath, a journal subdirectory is designed by MongoDB.

**13. How can you isolate our cursors from intervening with the write operations?**

Snapshot () method is used to isolate cursors from intervening with writes. This method negotiates the index and makes sure that each query comes to any article only once.

**14. Define MongoDB.**

It is document oriented database which is used to high availability, easy scalability and high performance. It supports the dynamic schema design.

**15. Explain replica set.**

It is a group of mongo instances that maintain same data set. Replica sets provide redundancy and high availability, and are the basis for all production deployments.

**16. What are the key features of mongodb?**

There are 3 main features of mongodb that are automatic scaling, High performance and high availability.

**17. What is CRUD?**

Mongodb provides CRUD operations that are create, Read, Update, Delete.

Click here to learn more about MongoDB CRUD operations in [MongoDB Training](https://intellipaat.com/mongodb-training/).

**18. What is sharding?**

Sharding means to store the data on the multiple machines.

**19. What is Aggregation in MongoDB?**

Aggregations are operations that process data records and return computed results.

**20. Define the namespace in mongodb?**

It is the concatenation of collection name and database.

**21. Which syntax is used to create collection in mongodb?**

db.createCollection(name,options) is used to create collection in mongodb.

**22. Which syntax is used to drop collection in mongodb?**

db.collection.drop() is used to drop collection in mongodb.

**23. Explain Replication.**

Replication is the process of synchronizing data across multiple servers.

**24. What is the use of Index in mongodb?**

Indexes provide high performance read operations for frequently used queries.

If you have any doubts or Queries related to MongoDB, get it clarifies from MongoDB Experts on [MongoDB Community](https://intellipaat.com/community/search?q=MongoDB).

**25. Which command is used for inserting a document?**

database.collection.insert (document) is used for inserting a document.

**26. What is use of GridFS in mongodb?**

GridFS is used for storing and retrieving the large files like audio, Images, Video files.

**27. What is the use journaling?**

Journaling is used to safe backups in mongodb.

**28. Which command is used to see the connection?**

db\_adminCommand (“connPoolStats”); is used to see the connection.

**29. Define primary replica sets?**

The primary replica set accepts all write operations from clients.

**30. Define secondary replica sets.**

The secondaries (page 565) replicate the primary’s oplog and apply the operations to their data sets such that the secondaries’ data sets reflect the primary’s data set.

**31. What is the use of profiler?**

Profiler is used to show the performance characteristics of every operation against the database.

**32. Which type of data mongodb store?**

MongoDB stores data in the form of documents, which are JSON-like field and value pairs.

**33. What is purpose of replication?**

Replication provides redundancy and increases data availability.

**34. what is embedded documents?**

Embedded documents capture relationships between data by storing related data in a single document structure.

**35. Define application level encryption.**

Application Level Encryption provides encryption on a per-field or per-document basis within the application layer.

**36. What is storage encryption?**

Storage Encryption encrypts all MongoDB data on the storage or operating system to ensure that only authorized processes can access protected data.

**37. Which method is used to create an index?**

CreateIndex() method is used to create an index.

**38. What is replica set oplog?**

The oplog records all operations that modify the data in the replica set.

**39. What is vertical scaling?**

Vertical scaling adds more CPU and storage resources to increase capacity.

**40. Define horizontal scaling.**

It divides the data set and distributes the data over multiple servers, or shards.

**41. What are the components of shared cluster?**

Sharded cluster has the following components: shards, query routers and config servers.

**42. Which command is use to create database?**

DATABASE\_NAME command is used to create database.

**43. Which command is use to drop database?**

db.dropDatabse() command is used to drop the database.

**44. What is the use of pretty() method?**

Pretty() method is used to show the results in a formatted way.

**45. Which method is used to remove the document form the collection?**

Remove() method is used to remove the document form the collection.

**46. Define Mongodb projection.**

Projection is used to select only necessary data.It did not select whole data of a document.

**47. What is the use of limit() method?**

Limit() method is used to limit the records in database.

**48. What is the syntax of limit() method?**

>db.COLLECTION\_NAME.find().limit(NUMBER) syntax is used.

**49. What is the syntax of sort() method?**

>db.COLLECTION\_NAME.find().sort({KEY:1}) syntax is used for sorting the documents.

**50. Which command is used to create the backup of database?**

Mongodump command is used to create the backup of database.

**51. What is collection in monodb?**

In monogodb collection is a group of mongodb documents.

**52. What is the use of db command?**

Db command gives the name of currently selected database.

**53. Which method is used to update the documents into a collection?**

Update() and save() methods are used to update the documents into a collection.

**54. What is the syntax of skip() method?**

The syntax of skip methopd is >db.COLLECTION\_NAME.find().limit(NUMBER).skip(NUMBER).

**55. Which command is used to restore the backup?**

Mongorestore command is used to restore the backup.

**56. What is the use of Dot notation in mogodb?**

MongoDB uses the dot notation to access the elements of an array and to access the fields of an embedded document.

**57. Define auditing.**

Auditing provides administrators with the ability to verify that the implemented security policies are controlling activity in the system.

**58. Define Aggregation Pipeline.**

It is a framework for performing aggregation tasks.T he pipeline is used to transform the documents into aggregated results.

**59. Define Map reduce.**

Map-reduce is a generic multi-phase data aggregation modality which is used for processing quantities of data.

**60. What is splitting in mongodb?**

It is a background process that is used to keep chunks from growing too large.

[](https://intellipaat.com/all-courses/big-data/?utm_source=salesforce-tutorial-page&amp;utm_medium=Tutorial-Page&amp;utm_campaign=May-all-courses/big-data/)

**61. Which language is used to wite mongodb?**

C++ language is used for writing and implementing mongodb.

**62. In which format mongodb stores the data?**

Mongodb uses collection to store the data rather than in table.

**63. What is the use of save() method?**

Save() method is used to replace the existing document to the new document.

**64. What is MongoDB?**

MongoDB (from humongous) is a cross-platform document-oriented database. Classified as a NoSQL database, MongoDB eschews the traditional table-based relational database structure in favor of JSON-like documents with dynamic schemas (MongoDB calls the format BSON), making the integration of data in certain types of applications easier and faster. Released under a combination of the GNU Affero General Public License and the Apache License, MongoDB is free and open-source software.

First developed by the software company 10gen (now MongoDB Inc.) in October 2007 as a component of a planned platform as a service product, the company shifted to an open source development model in 2009, with 10gen offering commercial support and other services. Since then, MongoDB has been adopted as backend software by a number of major websites and services, including Craigslist, eBay, Foursquare, SourceForge, Viacom, and the New York Times, among others. MongoDB is the most popular NoSQL database system.

For better understanding of MongoDB refer this [What is MongoDB](https://intellipaat.com/blog/what-is-mongodb/) blog.

**65. What is the use of MongoDB?**

MongoDB is relational database management system (RDBMS) replacement for web applications. So when you have something which is close to RDBMS, MongoDB could be of good use. It gives you that additional partition tolerance which RDMBS doesn’t give but it has problems with availability. But if you want more scalability, MongoDB would be your choice.

It’s suitable for real-time analytics and high speed logging. It’s highly scalable as well. Craigslist uses MongoDB for archived posts.

**66. What do you understand by NoSQL databases? Is MongoDB a NoSQL database? Explain.**

At the present time, the internet is loaded with big data, big users, big complexity etc. and also becoming more complex day by day. NoSQL is answer of all these problems; it is not a traditional database management system, not even a relational database management system (RDBMS). NoSQL stands for “Not Only SQL”. NoSQL is a type of database that can handle and sort all type of unstructured, messy and complicated data. It is just a new way to think about the database.Yes. MongoDB is a NoSQL database.

**67. What type of DBMS is MongoDB?**

MongoDB is a document oriented DBMS.

**68. What is the difference between MongoDB and MySQL?**

Although MongoDB and MySQL both are free and open source databases, there is a lot of difference between them in the term of data representation, relationship, transaction, querying data, schema design and definition, performance speed, normalization and many more. To compare MySQL with MongoDB is like a comparison between Relational and Non-relational databases.

**69. What is the use of MongoDB?**

* MongoDB is typically used as the primary data store for operational applications with real-time requirements (i.e., low-latency, high availability). MongoDB is generally a good fit for 60%-80% of the applications you may be building today. MongoDB is easy to operate and scale in ways that are hard if not impossible with relational databases.
* MongoDB excels in many use cases where relational databases aren’t a good fit, like applications with unstructured, semi-structured and polymorphic data, as well as applications with large scalability requirements or multi-data center deployments.
* MongoDB may not be a good fit for some applications. For example, applications that require complex transactions (e.g., a double-entry bookkeeping system) and scan-oriented applications that access large subsets of the data most of the time may not be a good fit for MongoDB. MongoDB is not a drop-in replacement for legacy applications built around the relational data model and SQL.
* Some common use cases include mobile apps, product catalogs, real-time personalization, content management and applications delivering a single view across multiple systems

**70. What kind of database is MongoDB?**

MongoDB is a document-oriented DBMS. Think of MySQL but with JSON-like objects comprising the data model, rather than RDBMS tables. Significantly, MongoDB supports neither joins nor transactions. However, it features secondary indexes, an expressive query language, atomic writes on a per-document level, and fully-consistent reads.  
Operationally, MongoDB features master-slave replication with automated failover and built-in horizontal scaling via automated range-based partitioning.

To learn more about MongoDB check [*MongoDB tutorial*](https://intellipaat.com/tutorial/mongodb-tutorial/).

**71. What language is MongoDB written in?**

MongoDB is implemented in C++. Drivers and client libraries are typically written in their respective languages, although some drivers use C extensions for better performance.

**72. What are the limitations of 32-bit versions of MongoDB?**

MongoDB uses memory-mapped files. When running a 32-bit build of MongoDB, the total storage size for the server, including data and indexes, is 2 gigabytes. For this reason, do not deploy MongoDB to production on 32-bit machines.  
If you’re running a 64-bit build of MongoDB, there’s virtually no limit to storage size. For production deployments, 64-bit builds and operating systems are strongly recommended.

**73. While creating Schema in MongoDB what are the points need to be taken in consideration?**

Points need to be taken in consideration are:

* Design your schema according to user requirements
* Combine objects into one document if you use them together. Otherwise, separate them
* Do joins while write, and not when it is on read
* For most frequent use cases optimize your schema
* Do complex aggregation in the schemaThis blog will help you get a better understanding of Difference Between Cassandra and MongoDB!

**1) Explain what is MongoDB?**

Mongo-DB is a document database which provides high performance, high availability and easy scalability.

**2) What is “Namespace” in MongoDB?**

MongoDB stores BSON (Binary Interchange and Structure Object Notation) objects in the collection. The concatenation of the collection name and database name is called a namespace.

**3) What is sharding in MongoDB?**

The procedure of storing data records across multiple machines is referred as Sharding. It is a MongoDB approach to meet the demands of data growth. It is the horizontal partition of data in a database or search engine. Each partition is referred as shard or database shard.

**4) How can you see the connection used by Mongos?**

To see the connection used by Mongos use db\_adminCommand (“connPoolStats”);

**5) Explain what is a replica set?**

A replica set is a group of mongo instances that host the same data set. In replica set, one node is primary, and another is secondary. From primary to the secondary node all data replicates.

**6) How replication works in MongoDB?**

Across multiple servers, the process of synchronizing data is known as replication. It provides redundancy and increase data availability with multiple copies of data on different database server. Replication helps in protecting the database from the loss of a single server.

**7) While creating Schema in MongoDB what are the points need to be taken in consideration?**

Points need to be taken in consideration are

* Design your schema according to user requirements
* Combine objects into one document if you use them together. Otherwise, separate them
* Do joins while write, and not when it is on read
* For most frequent use cases optimize your schema
* Do complex aggregation in the schema

**8) What is the syntax to create a collection and to drop a collection in MongoDB?**

* Syntax to create collection in MongoDB is db.createCollection(name,options)
* Syntax to drop collection in MongoDB is db.collection.drop()

**9) Explain what is the role of profiler in MongoDB?**

MongoDB database profiler shows performance characteristics of each operation against the database. You can find queries using the profiler that are slower than they should be.

**10) Explain can you move old files in the moveChunk directory?**

Yes, it is possible to move old files in the moveChunk directory, during normal shard balancing operations these files are made as backups and can be deleted once the operations are done.

**11) To do safe backups what is the feature in MongoDB that you can use?**

Journaling is the feature in MongoDB that you can use to do safe backups.

**12) Mention what is Objecld composed of?**

Objectld is composed of

* Timestamp
* Client machine ID
* Client process ID
* 3 byte incremented counter

**13) Mention what is the command syntax for inserting a document?**

For inserting a document command syntax is database.collection.insert (document).

**14) Mention how you can inspect the source code of a function?**

To inspect a source code of a function, without any parentheses, the function must be invoked.

**15) What is the command syntax that tells you whether you are on the master server or not? And how many master does MongoDB allow?**

Command syntax Db.isMaster() will tell you whether you are on the master server or not. MongoDB allows only one master server, while couchDB allows multiple masters.

**16) Mention the command syntax that is used to view Mongo is using the link?**

The command syntax that is used to view mongo is using the link is db.\_adminCommand(“connPoolStats.”)

**17) Explain what are indexes in MongoDB?**

Indexes are special structures in MongoDB, which stores a small portion of the data set in an easy to traverse form. Ordered by the value of the field specified in the index, the index stores the value of a specific field or set of fields.

**18) Mention what is the basic syntax to use index in MongoDB?**

The basic syntax to use in MongoDB is >db.COLLECTION\_NAME.ensureIndex ( {KEY:1} ). In here the key is the the name of the COLUMN (or KEY:VALUE pair) which is present in the documents.

**19) Explain what is GridFS in MongoDB?**

For storing and retrieving large files such as images, video files and audio files GridFS is used. By default, it uses two files fs.files and fs.chunks to store the file’s metadata and the chunks.

**20) What are alternatives to MongoDB?**

Cassandra, CouchDB, Redis, Riak,[Hbase](https://www.guru99.com/hbase-tutorials.html)are a few good alternatives.

Here’s a list of frequently asked MongoDB interview questions and answers for beginners as well as professionals.

**What are the best features of MongoDB?**

|  |  |
| --- | --- |
| **MongoDB Features** | |
| **Feature** | **Description** |
| ***Indexing*** | It indexes are created in order to improve the search performance. |
| ***Replication*** | MongoDB distributes the data across different machines. |
| ***Ad-hoc Queries*** | It supports ad-hoc queries by indexing the BSON documents & using a unique query language. |
| ***Schemaless*** | It is very flexible because of its schema-less database that is written in C++. |
| ***Sharding*** | MongoDB uses sharding to enable deployments with very large data sets and high throughput operations. |

**When using replication, can some members use journaling and others not?**

Yes!

**Can journaling feature be used to perform safe hot backups?**

Yes!

**What is 32-bit nuances?**

There is an extra memory mapped file activity with journaling. This will further constrain the limited db size of 32-bit builds. For now, journaling by default is disabled on 32-bit systems.

**Will there be journal replay programs in case of incomplete entries (if there is a failure in the middle of one)?**

Each journal (group) write is consistent and won’t be replayed during recovery unless it is complete.

**What is the role of profiler in MongoDB?**

MongoDB includes a database profiler which shows performance characteristics of each operation against the database. With this profiler you can find queries (and write operations) which are slower than they should be and use this information for determining when an index is needed.

**What is a ‘namespace’?**

MongoDB stores BSON objects in collections. The concatenation of the database name and the collection name (with a period in between) is called a ‘namespace’.

**When an object attribute is removed, is it deleted from the store?**

Yes, you can remove the attribute and then re-save() the object.

**Are null values allowed?**

[[](https://www.edureka.co/mongodb)](https://www.edureka.co/mongodb" \t "_blank)

**[MongoDB Certification Training](https://www.edureka.co/mongodb" \t "_blank)**

* *[Instructor-led Sessions](https://www.edureka.co/mongodb" \t "_blank)*
* *[Real-life Case Studies](https://www.edureka.co/mongodb" \t "_blank)*
* *[Assignments](https://www.edureka.co/mongodb" \t "_blank)*
* *[Lifetime Access](https://www.edureka.co/mongodb" \t "_blank)*

[Explore Curriculum](https://www.edureka.co/mongodb" \t "_blank)

Yes, but only for the members of an object. A null cannot be added to the database collection as it isn’t an object. But {}can be added.

**Does an update fsync to disk immediately?**

No. Writes to disk are lazy by default. A write may only hit the disk a couple of seconds later. For example, if the database receives thousand increments to an object within one second, it will only be flushed to disk once.*(Note: fsync options are available both at the command line and via getLastError\_old.)*

**How do I do transactions/locking?**

MongoDB does not use traditional locking or complex transactions with rollback, as it is designed to be light weight, fast and predictable in its performance. It can be thought of how analogous is to the MySQL’s MyISAM autocommit model. By keeping transaction support extremely simple, performance is enhanced, especially in a system that may run across many servers.

**Why are data files so large?**

MongoDB does aggressive preallocation of reserved space to avoid file system fragmentation.

**How long does replica set failover take?**

It may take 10-30 seconds for the primary to be declared down by the other members and a new primary to be elected. During this window of time, the cluster is down for primary operations i.e writes and strong consistent reads. However, eventually consistent queries may be executed to secondaries at any time (in slaveOk mode), including during this window.

**What’s a Master or Primary?**

This is a node/member which is currently the primary and processes all writes for the replica set. During a failover event in a replica set, a different member can become primary.

**What’s a Secondary or Slave?**

A secondary is a node/member which applies operations from the current primary. This is done by tailing the replication oplog (local.oplog.rs). Replication from primary to secondary is asynchronous, however, the secondary will try to stay as close to current as possible (often this is just a few milliseconds on a LAN).

**Is it required to call ‘getLastError’ to make a write durable?**

No. If ‘getLastError’ (aka ‘Safe Mode’) is not called, the server does exactly behave the way as if it has been called. The ‘getLastError’ call simply allows one to get a confirmation that the write operation was successfully committed. Of course, often you will want that confirmation, but the safety of the write and its durability is independent.

**Should you start out with Sharded or with a Non-Sharded MongoDB environment?**

[[](https://www.edureka.co/mongodb)](https://www.edureka.co/mongodb" \t "_blank)

**[MongoDB Certification Training](https://www.edureka.co/mongodb" \t "_blank)**

[Watch The Course Preview](https://www.edureka.co/mongodb" \t "_blank)

We suggest starting with Non-Sharded for simplicity and quick startup, unless your initial data set will not fit on single servers. Upgrading to Sharded from Non-sharded is easy and seamless, so there is not a lot of advantage in setting up Sharding before your data set is large.

**How does Sharding work with replication?**

Each Shard is a logical collection of partitioned data. The shard could consist of a single server or a cluster of replicas. Using a replica set for each Shard is highly recommended.

**When will data be on more than one Shard?**

MongoDB Sharding is range-based. So all the objects in a collection lie into a chunk. Only when there is more than 1 chunk there is an option for multiple Shards to get data. Right now, the default chunk size is 64mb, so you need at least 64mb for migration.

**What happens when a document is updated on a chunk that is being migrated?**

The update will go through immediately on the old Shard and then the change will be replicated to the new Shard before ownership transfers.

**What happens when a Shard is down or slow when querying?**

If a Shard is down, the query will return an error unless the ‘Partial’ query options is set. If a shard is responding slowly, Mongos will wait for it.

**Can the old files in the ‘moveChunk’ directory be removed?**

Yes, these files are made as backups during normal Shard balancing operations. Once the operations are done then they can be deleted. The clean-up process is currently manual so this needs to be taken care of to free up space.

**How do you see the connections used by Mongos?**

The following command needs to be used: db.\_adminCommand(“connPoolStats”);

**If a ‘moveChunk’ fails, is it necessary to cleanup the partially moved docs?**

No, chunk moves are consistent and deterministic. The move will retry and when completed, the data will be only on the new Shard.

**What are the disadvantages of MongoDB?**

* A 32-bit edition has 2GB data limit. After that it will corrupt the entire DB, including the existing data. A 64-bit edition won’t suffer from this bug/feature.
* Default installation of MongoDB has asynchronous and batch commits turned on. Meaning, it lies when asked to store something in DB and commits all changes in a batch at a later time in future. If there is a server crash or power failure, all those commits buffered in memory will be lost. This functionality can be disabled, but then it will perform as good as or worse than MySQL.
* MongoDB is only ideal for implementing things like analytics/caching where impact of small data loss is negligible.
* In MongoDB, it’s difficult to represent relationships between data so you end up doing that manually by creating another table to represent the relationship between rows in two or more tables.

**Q1) What’s a good way to get a list of all unique tags for a collection of documents millions of items large when we doesn't have access to mongodb’s new “distinct” command ?**

**Ans:** The normal way of doing tagging seems to be indexing multikeys. Even if your MongoDB driver doesn’t implement distinct, we can implement.

In JavaScript you can write something like this:

|  |  |
| --- | --- |
| 1 | result = db.$cmd.findOne({“distinct” : “collection\_name”, “key” : “tags”}) |

 You do a findOne on the “$cmd” collection of whatever database you’re using. Pass it the collection name and the key you want to run distinct on.

If you ever need a command your driver doesn’t provide a helper for, you can look at the below link for complete list of database commands.

https://docs.mongodb.com/manual/reference/command/

**Q2) How to get the names of all the keys in a MongoDB collection?**

**Ans:**

|  |  |
| --- | --- |
| 1  2  3  4  5  6 | For example, from this:  db.things.insert( { type : [‘dog’, ‘cat’] } );  db.things.insert( { egg : [‘cat’] } );  db.things.insert( { type : [] } );  db.things.insert( { hello : []  } );  How to get the unique keys: type, egg, hello |

We could do this with MapReduce:

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8 | mr = db.runCommand({  “mapreduce” : “my\_collection”,  “map” : function() {  for (var key in this) { emit(key, null); }  },  “reduce” : function(key, stuff) { return null; },  “out”: “my\_collection” + “\_keys”  }) |

Then run distinct on the resulting collection so as to find all the keys:

|  |  |
| --- | --- |
| 1  2 | db[mr.result].distinct(“\_id”)  [“foo”, “bar”, “baz”, “\_id”, …] |

**Q3) How to browse or query live MongoDB data?**

**Ans:**Below is some utilities available in the market.

1. MongoHub is moved to a native mac version, please check  https://github.com/bububa/MongoHub-Mac.

2. https://github.com/Imaginea/mViewer

 This is awesome with tree and document views.  
   
3. genghisapp

It is a web-based GUI that is clean, light-weight, straight-forward, offers keyboard shortcuts, and works awesomely. It also supports GridFS.  
Best of all, it’s a single script.

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9 | To install it              $ gem install genghisapp bson\_ext              (bson\_ext is optional but will greatly improve the performance of the gui)                To run it (this will automatically open your web browser and navigate to the app as well)              genghisapp                To stop it              genghisapp –kill |

**Q4) How to use map/reduce to handle more than 10000 unique keys for grouping in MongoDB?**

**Ans:**Use Version 2.2. The db.collection.group() method's returned array can contain at most 20,000 elements; i.e. at most 20,000 unique groupings. For group by operations that results in more than 20,000 unique groupings, use mapReduce. Previous versions had a limit of 10,000 elements.

**Q5) Can anyone share some insight on the index/RAM relationship and what happens when both an individual index and all of my indexes exceed the size of available RAM?**

**Ans:**MongoDB keeps what it can of the indexes in RAM. They’ll be swaped out on an LRU basis. You’ll often see documentation that suggests you should keep your “working set” in memory: if the portions of index you’re actually accessing fit in memory, you’ll be fine.

It is the working set size plus MongoDB’s indexes which should ideally reside in RAM at all times i.e. the amount of available RAM should ideally be at least the working set size plus the size of indexes plus what the rest of the OS (Operating System) and other software running on the same machine needs. If the available RAM is less than that, LRUing is what happens and we might therefore get significant slowdown.

One thing to keep in mind is that in an index btree buckets are cached, not individual index keys i.e. if we had a uniform distribution of keys in an index including for historical data, we might need more of the index in RAM compared to when we have a compound index on time plus something else. With the latter, keys in the same btree bucket are usually from the same time era, so this caveat does not happen. Also, we should keep in mind that our field names in BSON are stored in the records (but not the index) so if we are under memory pressure they should be kept short.

**Q10) What does an appropriate db.ClockTime.update() statement look like to convert these text based values to a date datatype?**

**Ans:**This code should do it:

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11 | > var cursor = db.ClockTime.find()  > while (cursor.hasNext()) {  … var doc = cursor.next();  … db.ClockTime.update({\_id : doc.\_id}, {$set : {ClockInTime : new Date(doc.ClockInTime)}})  … }  I have exactly the same situation as Jeff Fritz.  In my case I have succeed with the following simpler solution:  db.ClockTime.find().forEach(function(doc) {  doc.ClockInTime=new Date(doc.ClockInTime);  db.ClockTime.save(doc);  }) |

**Q11) Updating a specific key/value inside of an array field with MongoDB**

**Ans:**As a preface, I’ve been working with MongoDB for about a week now, so this may turn out to be a pretty simple answer.

I have data already stored in my collection, we will call this collection content, as it contains articles, news, etc. Each of these articles contains another array called author which has all of the author’s information (Address, Phone, Title, etc).

**The Goal –** I am trying to create a query that will update the author’s address on every article that the specific author exists in, and only the specified author block (not others that exist within the array).  
Sort of a “Global Update” to a specific article that affects his/her information on every piece of content that exists.  
Here is an example of what the content with the author looks like.

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25  26  27  28  29  30  31  32  33  34  35  36  37  38  39  40  41  42  43  44  45  46  47  48  49 | {  “\_id” : ObjectId(“4c1a5a948ead0e4d09010000”),  “authors” : [  {  “user\_id” : null,  “slug” : “joe-somebody”,  “display\_name” : “Joe Somebody”,  “display\_title” : “Contributing Writer”,  “display\_company\_name” : null,  “email” : null,  “phone” : null,  “fax” : null,  “address” : null,  “address2” : null,  “city” : null,  “state” : null,  “zip” : null,  “country” : null,  “image” : null,  “url” : null,  “blurb” : null  },  {  “user\_id” : null,  “slug” : “jane-somebody”,  “display\_name” : “Jane Somebody”,  “display\_title” : “Editor”,  “display\_company\_name” : null,  “email” : null,  “phone” : null,  “fax” : null,  “address” : null,  “address2” : null,  “city” : null,  “state” : null,  “zip” : null,  “country” : null,  “image” : null,  “url” : null,  “blurb” : null  },  ],  “tags” : [  “tag1”,  “tag2”,  “tag3”  ],  “title” : “Title of the Article”  } |

I can find every article that this author has created by running the following command:

db.content.find({authors: {$elemMatch: {slug: ‘joe-somebody’}}});

So theoretically I should be able to update the authors record for the slug joe-somebody but notjane-somebody (the 2nd author), I am just unsure exactly how you reach in and update every record for that author.

I thought I was on the right track, and here’s what I’ve tried.

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25  26  27  28  29  30  31  32  33  34  35  36  37  38  39  40  41  42  43  44  45  46  47  48  49 | {  “\_id” : ObjectId(“4c1a5a948ead0e4d09010000”),  “authors” : [  {  “user\_id” : null,  “slug” : “joe-somebody”,  “display\_name” : “Joe Somebody”,  “display\_title” : “Contributing Writer”,  “display\_company\_name” : null,  “email” : null,  “phone” : null,  “fax” : null,  “address” : null,  “address2” : null,  “city” : null,  “state” : null,  “zip” : null,  “country” : null,  “image” : null,  “url” : null,  “blurb” : null  },  {  “user\_id” : null,  “slug” : “jane-somebody”,  “display\_name” : “Jane Somebody”,  “display\_title” : “Editor”,  “display\_company\_name” : null,  “email” : null,  “phone” : null,  “fax” : null,  “address” : null,  “address2” : null,  “city” : null,  “state” : null,  “zip” : null,  “country” : null,  “image” : null,  “url” : null,  “blurb” : null  },  ],  “tags” : [  “tag1”,  “tag2”,  “tag3”  ],  “title” : “Title of the Article”  } |

Solution:  
This is what finally worked for me!

db.content.update({‘authors.slug’:’joe-somebody’},{$set:{‘authors.$.address’:’Address That I wanted’}},false,true);

It updates all the records properly, thanks!  
Maybe you can use the $ operator (positional-operator)?

Explore MongoDB Sample Resumes! Download & Edit, Get Noticed by Top Employers! [**Download Now!**](https://mindmajix.com/mongo-db-sample-resumes)

**Q12) MongoDB – simulate join or subquery**

**Ans:**I’m trying to figure out the best way to structure my data in Mongo to simulate what would be a simple join or subquery in SQL.

Say I have the classic Users and Posts example, with Users in one collection and Posts in another. I want to find all posts by users who’s city is “london”.

I’ve simplified things in this question, in my real world scenario storing Posts as an array in the User document won’t work as I have 1,000’s of “posts” per user constantly inserting.

**Q13) Can Mongos $in operator help here? Can $in handle an array of 10,000,000 entries?**

**Ans:**Honestly, if you can’t fit “Posts” into “Users”, then you have two options.  
1. Denormalize some User data inside of posts. Then you can search through just the one collection.  
2. Do two queries. (one to find users the other find posts)  
Based on your question, you’re trying to do #2.  
Theoretically, you could build a list of User IDs (or refs) and then find all Posts belonging to a User$in that array. But obviously that approach is limited.

**Q14) Can $in handle an array of 10,000,000 entries?**

**Ans:**Look, if you’re planning to “query” your posts for all users in a set of 10,000,000 Users you are well past the stage of “query”. You say yourself that each User has 1,000s of posts so you’re talking about a query for “Users with Posts who live in London” returning 100Ms of records.  
**100M records isn’t a query, that’s a dataset!**

If you’re worried about breaking the $in command, then I highly suggest that you use map/reduce. The Mongo Map/Reduce will create a new collection for you. You can then trim down or summarize this dataset as you see fit.

$in can handle 100,000 entries. I’ve never tried 10,000,000 entries but the query (a query is also a document) has to be smaller than 4mb (like every document) so 10,0000,0000 entries isn’t possible.  
Why don’t you include the user and its town in the Posts collection? You can index this town because you can index properties of embedded entities. You no longer have to simulate a join because you can query the Posts on the towns of its embedded users.

This means that you have to update the Posts when the town of a user changes but that doesn’t happen very often. This update will be fast if you index the UserId in the Posts collection.

**Q15) Mongo complex sorting?**

**Ans:**I know how to sort queries in MongoDB by multiple fields, e.g.,db.coll.find().sort({a:1,b:-1}).

**Q16) Can I sort with a user-defined function; e.g., supposing a and b are integers, by the difference between a and b (a-b)?**

**Ans:**I don’t think this is possible directly; the sort documentation certainly doesn’t mention any way to provide a custom compare function.

You’re probably best off doing the sort in the client, but if you’re really determined to do it on the server you might be able to use db.eval() to arrange to run the sort on the server (if your client supports it).

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11 | Server-side sort:  db.eval(function() {  return db.scratch.find().toArray().sort(function(doc1, doc2) {  return doc1.a – doc2.a  })  });    Versus the equivalent client-side sort:  db.scratch.find().toArray().sort(function(doc1, doc2) {  return doc1.a – doc2.b  }); |

Note that it’s also possible to sort via an aggregation pipeline and by the $orderby operator (i.e. in addition to .sort()) however neither of these ways lets you provide a custom sort function either. Why don’t create the field with this operation and sort on it ?

**Related Page:**[**MongoDB Vs PostgreSQL Comparison**](https://mindmajix.com/mongodb-vs-postgresql)

**Q17) How to set a primary key in MongoDB?**

**Ans:**I want to set one of**my fields as the primary key.**I am using MongoDB as my NoSQL.  
The \_id field is reserved for primary key in mongodb, and that should be an unique value. If you don’t set anything to \_id it will automatically fill it with “MongoDB Id Object”. But you can put any unique info into that field.

Additional info: <a href="https://www.mongodb.org/display/DOCS/BSON">https://www.mongodb.org/display/DOCS/BSON</a>

Hope it helps.

**Q18) Delete everything in a MongoDB database**

**Ans:**I’m doing development on MongoDB. For totally non-evil purposes, I sometimes want to blow away everything in a databasethat is, to delete every single collection, and whatever else might be lying around, and start from scratch. Is there a single line of code that will let me do this? Bonus points for giving both a MongoDB console method and a MongoDB Ruby driver method.

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10 | use [database];  db.dropDatabase();  Ruby code should be pretty similar.  Also, from the command line:  mongo [database] –eval “db.dropDatabase();”Use  [databaseName]  db.Drop+databaseName();  drop collection  use databaseName  db.collectionName.drop(); |

**Q19) How to $set sub-sub-array items in MongoDB**

**Ans:**I’m developing a webapp which has a portal-ish component to it (think like multiple panels that can be drug around from column to column and added or removed). I’m using MongoDB to store this info with a format like so…  
Skip code block

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25  26  27  28  29 | {  \_id: ObjectId(…),  title: ‘My Layout’,  columns: [  {  order: 1,  width: 30,  panels: [  { title: ‘Panel Title’, top: 100, content: ‘…’ },  { title: ‘Panel Title’, top: 250, content: ‘…’ },  ]  },  {  … multiple columns …  }  ]  }    I’m attempting to use atomic/modifier operations with update() and this is getting confusing. If I wanted to just update one property of a specific panel, how do I reference that?  update(  { \_id: ObjectId(…) },  { $set: { columns.[???].panels.[???].top: 500 }  )    If you know the index in the array you can access the array element directly using dot notation.  update(  { \_id: ObjectId(xxxx) },  { $set: { ‘columns.0.panels.0.top’ : 125}}  ) |

Make sure you encase the dot notated path in quotes as a string.

**Edit:**

To give more detail on how this could work dynamically, I’ll give an example in PHP:

$action = array(“columns.$colNum.panels.$panelNum” => $newValue);

Yes there is the positional operator, but it does not appear to be advanced enough to change arrays within arrays, this may change in MongoDB 1.7.0  
There is an alternative you can do instead of trying to stuff this information into a nested document. Try to flatten it out. You can create a collection that has panel & column objects:  
column object:

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16 | {  \_id: // MongoId  type: ‘column’,  user: ‘username’,  order: 1,  width: 30,  }  panel object:  {  \_id: //MongoId  type: ‘panel’,  user: ‘username’,  parentColumn: //the columns \_id string  top: 125,  left: 100  } |

Then you can find all columns that belong to a user by doing:

|  |  |
| --- | --- |
| 1  2  3  4  5 | find({ type: ‘column’, user:’username’});  You can find all panels for a specific column by doing:  find({type: ‘panel’, columnOwner:’ownerID’});  Since each column and panel will have a unique ID given by MongoDB to it, you can easily query and atomically set options.  update({‘\_id’: ObjectId(‘idstring’)}, {$set : { ‘top’ : 125}}); |

**Q20) Some beginner’s questions about MongoDB**

**Ans:**I’m a beginner with MongoDB and I’ve some questions:

1. When I’m connected to Mongo, and i executeshow dbs I see 2 databases: admin and local. What’s their role? Then if I execute an insert command likedb.foo.insert({“value”:”mongo”}), the test database appears. Why? How can i specify a custom name for a database?  
2. Withshow dbs I get the databases (somehow like show databases in sql), how can I then list the collections inside a database (I would use show tables in sql)?  
3. When executing a command, the MongoDB tutorial always uses thedb object. Is it the main object (a sort of “connection” object) that has to used for executing commands or it’s something else?  
Thanks!  
1. Adminand local contain various settings local to the server, like users who are authenticated to connect. Under beginner usage, you shouldn’t need to worry about them at all. By default you connect to a database named test. To connect to a new database, just use databasename from the mongo command line, or mongo databasename from your OS shell.  
2. Use [database\_name]and then show collections  
3. Thedb object is your root handle to the currently-selected database on the mongo commmand line. The command line is really just a Javascript command line, and there are various mongodb-specific objects and functions exposed that let you do stuff. Try help() for a full listing.

**Related Page:**[**MongoDB GUI - Top 7 MongoDB GUI Tools**](https://mindmajix.com/mongodb-gui-tools)

**Q21) How to update based on existing data in mongo**

**Ans:**Sort of a mongo noob, and I have a feeling I am tackling this problem from the wrong direction.  
I have about a 7 million document collection. Each document has two fields that I want to modify(not replace), basically they are big strings that have , and I want to replace those with n.  
I spent about an hour trying to find a way to “back reference” the object returned by the query, which totally doesn’t exist. What is the best approach for something like this?  
You’ll have to query for all the documents and update them one by one. If you do it in JavaScript, it would be something like:

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7 | mydb = db.getSisterDB(“whateverDBYoureUsing”);  var cursor = mydb.foo.find();  while (cursor.hasNext()) {  var x = cursor.next();  /\* replace with n in x’s strings … \*/  db.foo.update({\_id : x.\_id}, x);  } |

You can copy this into a .js file (say, replace.js), change the db and collection names, and run it as a script from the shell:  
mongo replace.js

**Q22) How to print out more than 20 items (documents) in MongoDB’s shell?**

**Ans:**

|  |  |
| --- | --- |
| 1  2  3  4 | db.foo.find().limit(300)  won’t do it… it still prints out 20  db.foo.find().toArray()  db.foo.find().forEach(printjson) |

will both print out very expanded view of each document instead of the 1-line version for find():  
DBQuery.shellBatchSize = 300  
will do.  
MongoDB Docs – Getting Started with the mongo Shell – Executing Queries  
You can use it inside of the shell to iterate over the next 20 results. Just type it if you see “has more” and you will see the next 20 items.  
from the shell if you want to show all results you could do db.collection.find().toArray() to get all results without it  
Could always do:

|  |  |
| --- | --- |
| 1  2  3  4 | db.foo.find().forEach(function(f){print(tojson(f, ”, true));});  To get that compact view.  Also, I find it very useful to limit the fields returned by the find so:  db.foo.find({},{name:1}).forEach(function(f){print(tojson(f, ”, true));}); |

which would return only the \_id and name field from foo.

**Q23) How to store timestamps? Are created and updated fields available automatically?**

**Ans:**What’s the best way to store timestamps in MongoDB?  
Which format is best:

|  |  |
| --- | --- |
| 1  2  3 | # “created”: { “d” : “2010-03-29”, “t” : “20:15:34” }  # “created”: “12343545234” # seconds since epoc  # “created”: “2010-03-14T21:20:14+0000” |

Is there a way to have MongoDB automatically setcreated and updated fields?  
Which format is best  
Best for what?  
Is there a way to have MongoDB automatically set created and updated fields?  
Created time is in the ObjectId but, as far as I know, you will have to update a updated field manually.

Example:

ObjectId(“538141a9615760fd04ffef5f”).getTimestamp()

1. The format you need to process it with best performance in your application should be preferred. Note that as default every document in MongoDB gets a created timestamp

(https://www.mongodb.org/display/DOCS/Object+IDs#ObjectIDs-DocumentTimestamps)

2. See 1) + I think you need to manually set the “update” field.  
If you do following on mongo shell it shows you time stamp that represents when that documents inserted using mongoId. For ex. ObjectId(“51f3dee5ee49f9b91e0db133”).getTimestamp(), then it returns ISODate.

**Q24) MongoDB: Updating documents using data from the same document**

**Ans:**I have a list of documents, each with lat and lon properties (among others).

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7 | { ‘lat’: 1, ‘lon’: 2, someotherdata […] }  { ‘lat’: 4, ‘lon’: 1, someotherdata […] }  […]  I want to modify it so that it looks like this:  { ‘coords’: {‘lat’: 1, ‘lon’: 2}, someotherdata […]}  { ‘coords’: {‘lat’: 4, ‘lon’: 1}, someotherdata […]}  […] |

So far I’ve got this:  
db.events.update({}, {$set : {‘coords’: {‘lat’: db.events.lat, ‘lon’: db.events.lon}}}, false, true)  
But it treats the db.events.lat and db.events.lon as strings. How can I reference the document’s properties?  
Cheers.  
The $rename operator (introduced a month after this question was posted) makes it really easy to do these kinds of things where you don’t need to modify the values.  
**Insert some test documents**  
db.events.insert({ ‘lat’: 1, ‘lon’: 2, someotherdata: [] })  
db.events.insert({ ‘lat’: 4, ‘lon’: 1, someotherdata: [] })  
**use the $rename operator**  
db.events.update({}, {$rename: {‘lat’: ‘coords.lat’, ‘lon’: ‘coords.lon’}}, false, true)  
**Results**

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18 | Skip code block  db.events.find()  {  “\_id” : ObjectId(“5113c82dd28c4e8b79971add”),  “coords” : {  “lat” : 1,  “lon” : 2  },  “someotherdata” : [ ]  }  {  “\_id” : ObjectId(“5113c82ed28c4e8b79971ade”),  “coords” : {  “lat” : 4,  “lon” : 1  },  “someotherdata” : [ ]  } |

**Update**: If all you have to do is change the structure of a document without changing the values, see gipset’s answer for a nice solution.  
According to a (now unavailable) comment on the Update documentation page, you cannot reference the current document’s properties from within an update().  
You’ll have to iterate through all the documents and update them like this:

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12 | Skip code block  db.events.find().snapshot().forEach(  function (e) {  // update document, using its own properties  e.coords = { lat: e.lat, lon: e.lon };  // remove old properties  delete e.lat;  delete e.lon;  // save the updated document  db.events.save(e);  }  ) |

Such a function can also be used in a map-reduce job or a server-side db.eval() job, depending on your needs.

**Q25) In MongoDB how do you use $set to update a nested value?**

**Ans:**In MongoDB how do you use $set to update a nested value?  
Using the dot notation:  
db.people.update({ }, { $set: { “address.street”: “Main Street” } })

**Q26) Mongorestore of a db causing me trouble**

**Ans:**I’m new to MongoDB and I have hard time to backup my local DB and restore it on my server. I found the link on Mongo’s website:

https://www.mongodb.org/display/DOCS/Import+Export+Tools but I still have problems with the restore.

When I do my backup I call  
mongodump –db Gen  
Then I see that all the collections are dump in /bin/dump/Gen folder  
I copy-paste from local to the server in the same folder the call  
mongorestore –db Gen –drop –dbpath dump/Gen  
But I get the following : Error : root directory must be a dump of a single database when specifying a db name with –db  
What am I doing wrong?  
Thanks for the help!  
Ok I find out what I’m doing wrong :  
I was doing  
mongorestore –db Gen –drop –dbpath dump/Gen  
But without the –dbpath it works just fine!  
mongorestore –db Gen –drop dump/Gen  
Thanks everyone!

**Q27) How to create user accounts in MongoDB?**

**Ans:**I wonder what’s ‘correct’ way to create user accounts in MongoDB and actions like register/login. Do I have to create a specific collection for users (Username,Email,Password) or MongoDB has something built in already for users?

If I have to create users collection manually, how to deal with password encryption? Thanks.

You’ll probably have to create and manage the collection of users manually.

As for encrypting passwords, the common approach is to hash the passwords using a suitable hash-function before you store them in the db. Later, when a user tries to login you use the same hash-function on the string they entered in the password field and compare that result to the password entry in your db. That way you never store the actual passwords and if someone hacks you they won’t get your users passwords, just their hashes.

For extra security (against dictionary attacks) you should also salt the hashed passwords. Read about it here

**Q28) How update the \_id of one MongoDB Document?**

I**Ans:** want update an \_id MongoDB of one document. I know it’s not a really good pratice. But with some technical reason, I need update it. But If I try to update it I have :

> db.clients.update({‘\_id’:ObjectId(“4cc45467c55f4d2d2a000002”)}, {‘$set’:{‘\_id’:ObjectId(“4c8a331bda76c559ef000004”)}});

Mod on \_id not allowed  
And the update is not made. How I can really update it ?  
You cannot update it. You’ll have to save the document using a new \_id, and then remove the old document.

Skip code block

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10 | // store the document in a variable  doc = db.clients.findOne({\_id: ObjectId(“4cc45467c55f4d2d2a000002”)})  // set a new \_id on the document  doc.\_id = ObjectId(“4c8a331bda76c559ef000004”)  // insert the document, using the new \_id  db.clients.insert(doc)  // remove the document with the old \_id  db.clients.remove({\_id: ObjectId(“4cc45467c55f4d2d2a000002”)})  To do it for your whole collection you can also use a loop (based on Niels example):  db.status.find().forEach(function(doc){ var id=doc.\_id; doc.\_id=doc.UserId; db.status.insert(doc); db.status.remove({\_id:id}); }) |

In this case UserId was the new ID I wanted to use

**Related Page:**[**MongoDB Query & Examples**](https://mindmajix.com/mongodb-query-and-examples)

**Q29) Get MongoDB Databases in a Javascript Array?**

I know that in the MongoDB terminal, I can run “show dbs” to see the available databases. I want to get them programmatically so that I can iterate over them and delete some based upon a regular expression.  
I have tried db.runCommand(“show dbs”) but that doesn’t work.  
Thanks in advance.

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10 | > db.getMongo().getDBNames()  [  “test”,  “admin”,  “local”  ]  > db.getMongo().getDBNames  function () {  return this.getDBs().databases.map(function (z) {return z.name;});  } |

Based upon this answer

<a href="https://groups.google.com/group/mongodb-user/browse\_thread/thread/9b3568f3a3cf4271,">https://groups.google.com/group/mongodb-user/browse\_thread/thread/9b3568f3a3cf4271, </a>

I was able to code up a solution.

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9 | use admin    dbs = db.runCommand({listDatabases: 1})  dbNames = []  for (var i in dbs.databases) { dbNames.push(dbs.databases[i].name) }  Hopefully this will help someone else.  The below will create an array of the names of the database:  var connection = new Mongo();  var dbNames = connection.getDBNames(); |

**Q31) MongoDB and “joins”**

**Ans:**I’m sure MongoDB doesn’t officially support “joins”. What does this mean? Does this mean “We cannot connect two collections(tables) together.”?  
I think if we put the value for \_id in collection A to the other\_id in collection B, can we simply connect two collections?  
If my understanding is correct, MongoDB can connect two tables together, say, when we run a query. This is done by “Reference” written in

<a href="https://www.mongodb.org/display/DOCS/Schema+Design">https://www.mongodb.org/display/DOCS/Schema+Design</a>.

Then what does “joins” really mean?  
I’d love to know the answer because this is essential to learn MongoDB schema design.

<a href="https://www.mongodb.org/display/DOCS/Schema+Design">https://www.mongodb.org/display/DOCS/Schema+Design</a>

It’s no join since the relationship will only be evaluated when needed. A join (in a SQL database) on the other hand will resolve relationships and return them as if they were a single table (you “join two tables into one”). You can read more about DBRef here:

<a href="https://docs.mongodb.org/manual/applications/database-references/">https://docs.mongodb.org/manual/applications/database-references/</a>

There are two possible solutions for resolving references. One is to do it manually, as you have almost described. Just save a document’s \_id in another document’s other\_id, then write your own function to resolve the relationship. The other solution is to use DBRefs as described on the manual page above, which will make MongoDB resolve the relationship client-side on demand. Which solution you choose does not matter so much because both methods will resolve the relationship client-side (note that a SQL database resolves joins on the server-side).

The database does not do joins — or automatic “linking” between documents. However you can do it yourself client side. If you need to do 2, that is ok, but if you had to do 2000, the number of client/server turnarounds would make the operation slow.

In MongoDB a common pattern is embedding. In relational when normalizing things get broken into parts. Often in mongo these pieces end up being a single document, so no join is needed anyway. But when one is needed, one does it client-side.

Consider the classic ORDER, ORDER-LINEITEM example. One order and 8 line items are 9 rows in relational; in MongoDB we would typically just model this as a single BSON document which is an order with an array of embedded line items. So in that case, the join issue does not arise. However the order would have a CUSTOMER which probably is a separate collection – the client could read the cust\_id from the order document, and then go fetch it as needed separately.

There are some videos and slides for schema design talks on the mongodb.org web site I belive.

The first example you link to shows how MongoDB references behave much like lazy loading not like a join. There isn’t a query there that’s happening on both collections, rather you query one and then you lookup items from another collection by reference.

**Mongo interface**

**Q31) What are some GUIs to use with Mongo, and what features do they offer? I’m looking for facts here, not opinions on which interface is best.**

**Ans:**

**Official List from MongoDB**

<a href="https://www.mongodb.org/display/DOCS/Admin+UIs">https://www.mongodb.org/display/DOCS/Admin+UIs</a>

**Web Based**

For PHP, I’d recommend Rock Mongo. Solid, lots of great features, easy setup.

<a href="https://rockmongo.com/">https://rockmongo.com/</a>

If you don’t want to install anything … you can use MongoHQ’s web interface (even if you your MongoDB isn’t on MongoHQ.)

<a href="https://mongohq.com/home">https://mongohq.com/home</a>

**Mac OS X**

While MongoHub had been a decent option for a while it’s bugs make it virtually unusable at this point …  
There is a more up-to-date (and less buggy) fork of the MongoHub project available:

https://github.com/fotonauts/MongoHub-Mac you can download a binary here.

**Windows**

By far, the best UI (for Windows) currently out there is MongoVUE.

<a href="https://blog.mongovue.com/">https://blog.mongovue.com/</a>

Looks great, lots of features, and if you are new it will really help you get going.

<a href="https://blog.mongovue.com/features/">https://blog.mongovue.com/features/</a>

Here’s a Q&A with the author too if you are interested

<a href="https://learnmongo.com/posts/qa-ishann-kumar-creator-of-mongovue/">https://learnmongo.com/posts/qa-ishann-kumar-creator-of-mongovue/</a>

On Mac there is MongoHub. On Windows you could try MongoVUE.  
Also see Do any visual tools exist for MongoDB (for Windows)?  
Screenshot of MongoHub:  
Here’s the official page of Admin UIs.  
I have not really used any of them. But it looks like there is quite a bit of coverage there.

**Web**

At the shop where I work we use the Prudence platform for some stuff, and also MongoDB, so we of course use MongoVision a lot. Browser based, tabbed collection views, pretty-printed document editor, and three themes OOB. Open source.

<a href="https://code.google.com/p/mongo-vision/">https://code.google.com/p/mongo-vision/</a>

**OS X**

MongoHub was as reliable as MongoVision.

https://mongohub.todayclose.com/

**Q32) Why not MongoDB?**

**Ans:**I recently used MongoDB for the first time and found it exceptionally easy to use and high-performing. Which leads to my question – why not MongoDB?  
Lets say I am implementing a Q & A app. My approach would be to implement the User data in a MySQL database and then use MongoDB for the question and answer storage – one collection storing a question and all responses.

**Q33) Is there anything wrong with this approach?**

**Ans:**MongoDB sounds like a fine application for your problem, but there are plenty of reasons why you would not use it.

MongoDB would not be well suited for applications that need:

1. Multi-Object Transactions: MongoDB only supports ACID transactions for a single document.
2. SQL: SQL is well-known and a lot of people know how to write very complex queries to do lots of things. This knowledge is transferrable across a lot of implementations where MongoDB’s queries language are specific to it.
3. Strong ACID guarantees: MongoDB allows for things like inconsistent reads which is fine in some applications, but not in all.
4. Traditional BI: A lot of very powerful tools exist that allow for OLAP and other strong BI applications and those run against traditional SQL database.

**Possible downsides:**

**Ans:**

1. You work in an organization that has only used SQL relational databases. You have no approval or support for using a NoSQL database yet.
2. You’ve never administered a MongoDB cluster; there’s a learning curve, as with all technologies.
3. Your data is really relational (e.g., one User has many Questions; a Question has many Answers), and you’ve overlooked the possibility.

MondoDB is a fine solution, a good alternative for those situations where it applies. If you can use it, why not?  
MongoDB is a brilliant database and I enjoy using it. That said, it has a few gotchas if you come from the world of SQL.  
Apart from ACID and other things that are well documented (and in other answers too), these things have caught us by surprise:

1. MongoDB expects you to have memory.**Lots of memory.** If you can’t fit your working set in memory, you can forget about it. This is different from most relational DBs which use memory only as cache! **To be more specific:**MongoDB uses RAM as primary storage and “swaps” the unneeded parts out to disk (Mongo leaves the decision over which parts get “swapped” to kernel). Traditional RDBMS work the other way around – they use disk as primary storage and use RAM as caching mechanism. So in general MongoDB uses more RAM. This is not a bad thing by itself, but as a consequence “real” RAM consumption is difficult to predict, which can lead to serious and unexpected degradation of performance once the working set grows over the (hard to predict) limit.

2. **Storage does not auto-shrink** when you remove records. The space that is allocated per collection stays allocated until you either repair DB or drop the collection. And it is allocated in huge chunks on a DB level (data files), which are then allocated to collections when needed (extents). That said, inside the collection’s allocated space the documents that are removed DO release their space for other documents in the same collection. This is a good explanation of concepts: <https://www.10gen.com/presentations/storage-engine-internals>

3. As a contrast to SQL which is parsed server-side, in Mongo you pass the data structures to query and CRUD functions. The consequence is that**each driver provides a different syntax**, which is a bit annoying. For instance, PyMongo uses a list of tuples instead of a dictionary (probably because dict in Python does not preserve order of keys) to specify which fields will be returned byfind(): (to be fair, that was probably the only sane way to do it – but it is a consequence of not using string-based language such as SQL)  
a.   MongoDB shell: db.test.find({}, {a:1})  
b.   PyMongo: db.find({}, fields=[(a,1,)]

This should not be viewed as a criticism of MongoDB – I enjoy using it and it has proven to be a reliable and performant tool. But to use it properly you need to learn about its space management.

**Q34) Track MongoDB performance?**

**Ans:**Is there a way to track ‘query’ performance in MongoDB? Specially testing indexes or subdocuments?  
In sql you can run queries, see execution time and other analytic metrics.  
I have a huge mongoDB collection and want to try different variations and indexes, not sure how to test it, would be nice to see how long did it take to find a record.. (I am new in MongoDB). Thanks  
There are two things here that you’ll likely be familiar with.  
1. Explain plans  
2. Slow Logs

**Explain Plans**

Here are some basic docs on explain. Running explain is as simple asdb.foo.find(query).explain(). (note that this actually runs the query, so if your query is slow this will be too)  
To understand the output, you’ll want to check some of the docs on the slow logs below. You’re basically given details about “how much index was scanned”, “how many are found”, etc. As is the case with such performance details, interpretation is really up to you. Read the docs above and below to point you in the right direction.

**Slow Logs**

By default, slow logs are active with a threshold of 100ms. Here’s a link to the full documentation on profiling. A couple of key points to get you started:  
Get/Set profiling:

|  |  |
| --- | --- |
| 1  2  3  4 | db.setProfilingLevel(2); // 0 => none, 1 => slow, 2 => all  db.getProfilingLevel();  See slow queries:  db.system.profile.find() |

**Q35) Mongodb can’t start**

**Ans:**today I updated my Mongo.. mongodb-stable (from 10gen repo)  
but my service has down. the following command not working

|  |  |
| --- | --- |
| 1  2 | $ sudo service mongodb start  $ start: Unknown job: mongodb |

even this command not working

|  |  |
| --- | --- |
| 1  2  3 | $ sudo /etc/init.d/mongodb start  $ Rather than invoking init scripts through /etc/init.d, use the service(8)  utility, e.g. service mongodb start |

Since the script you are attempting to invoke has been converted to an  
Upstart job, you may also use the start(8) utility, e.g. start mongodb  
start: Unknown job: mongodb  
there is no mongo process running

|  |  |
| --- | --- |
| 1  2 | $ ps -ef|grep mongo  $ user  9689  8121  0 13:01 pts/1    00:00:00 grep –color=auto mongo |

log is here  
Skip code block

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22 | tail /var/log/mongodb/mongodb.log  Fri Dec 10 11:24:35 [conn4] end connection 127.0.0.1:54217  Fri Dec 10 11:25:35 [initandlisten] connection accepted from 127.0.0.1:54229 #5  Fri Dec 10 11:26:25 [initandlisten] connection accepted from 127.0.0.1:54243 #6  Fri Dec 10 11:26:30 [conn6] end connection 127.0.0.1:54243  Fri Dec 10 11:30:13 got kill or ctrl c or hup signal 15 (Terminated), will terminate after current cmd ends  Fri Dec 10 11:30:13 [interruptThread] now exiting  Fri Dec 10 11:30:13 dbexit:  Fri Dec 10 11:30:13 [interruptThread] shutdown: going to close listening sockets…  Fri Dec 10 11:30:13 [interruptThread] closing listening socket: 5  Fri Dec 10 11:30:13 [interruptThread] closing listening socket: 6  Fri Dec 10 11:30:13 [interruptThread] closing listening socket: 7  Fri Dec 10 11:30:13 [interruptThread] closing listening socket: 8  Fri Dec 10 11:30:13 [interruptThread] shutdown: going to flush oplog…  Fri Dec 10 11:30:13 [interruptThread] shutdown: going to close sockets…  Fri Dec 10 11:30:13 [interruptThread] shutdown: waiting for fs preallocator…  Fri Dec 10 11:30:13 [interruptThread] shutdown: closing all files…  Fri Dec 10 11:30:13     closeAllFiles() finished  Fri Dec 10 11:30:13 [interruptThread] shutdown: removing fs lock…  Fri Dec 10 11:30:13 dbexit: really exiting now  for now, I’m running Mongo through this command just for a while, creating process manually  $ sudo mongod -f /etc/mongodb.conf |

any idea? or has anyone updated Mongo-stable via Update manager?

**Edits**:

This mongodb version was v1.6.5 and it seems mongo team released it as stable with a bug. And they fixed it immediately at v1.7.4. You can see the major priority issue.  
Bug has been reported and fixed.

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10 | https://jira.mongodb.org/browse/SERVER-2200  $ sudo apt-get purge mongodb-stable  $ sudo apt-get install mongodb-stable  (remove the lock file if present in /var/lib/mongodb)  $ sudo init 6  Then edit /etc/init/mongodb.conf removing the line “limit nofile 20000”  $ sudo vi /etc/init/mongodb.conf  $ sudo service mongodb start  mongodb start/running, process 2351  Worked. |

Running **Ubuntu 11.10**confirm you have the latest version of MongoDB:

|  |  |
| --- | --- |
| 1  2 | $ mongod –version  db version v2.2.0 |

(If you don’t have the latest version of MongoDB, follow the <strong>Ubuntu Installation instructions </strong>at <br><br><a href="https://docs.mongodb.org/manual/tutorial/install-mongodb-on-ubuntu/)"> https://docs.mongodb.org/manual/tutorial/install-mongodb-on-ubuntu/)</a>

First confirm that the **mongodb**user/group has permission to write to the data directory:

|  |  |
| --- | --- |
| 1  2  3 | $ sudo chown -R mongodb:mongodb /var/lib/mongodb/.  Start up MongoDB as a Daemon (background process) using the following command:  $ mongod –fork –dbpath /var/lib/mongodb/ –smallfiles –logpath /var/log/mongodb.log –logappend |

To Shut Down MongoDB enter the Mongo CLI, access the admin and issue the shutdown command:

|  |  |
| --- | --- |
| 1  2  3 | $ ./mongo  > use admin  > db.shutdownServer() |

See: https://www.mongodb.org/display/DOCS/Starting+and+Stopping+Mongo

**Q36) MongoDB: Unique Key in Embedded Document**

**Ans:**Is it possible to set a unique key for a key in an embedded document?  
I have a Users collection with the following sample documents:

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15 | Skip code block  {  Name: “Bob”,  Items: [  {  Name: “Milk”  },  {  Name: “Bread”  }  ]  },  {  Name: “Jim”  }, |

Is there a way to create an index on the property Items.Name?  
I got the following error when I tried to create an index:

> db.Users.ensureIndex({“Items.Name”: 1}, {unique:true});

E11000 duplicate key error index: GroceryGuruApp.Users.$Items.Name\_1  dup key: {  
: null }  
Any suggestions? Thank you!

Unique indexes exist only across collection. To enforce uniqueness and other constraints across document you must do it in client code. (Probably virtual collections would allow that, you could vote for it.)

What are you trying to do in your case is to create index on key Items.Name which doesn’t exist in any of the documents (it doesn’t refer to embedded documents inside array Items), thus it’s null and violates unique constraint across collection.

The index will be across all Users and since you asked it for ‘unique’, no user will be able to have two of the same named item AND no two users will be able to have the same named Item.  
Is that what you want?

Furthermore, it appears that it’s objecting to two Users having a ‘null’ value for Items.Name, clearly Jim does, is there another record like that?  
It would be unusual to require uniqueness on an indexed collection like this.

MongoDB does allow unique indexes where it indexes only the first of each value, see https://www.mongodb.org/display/DOCS/Indexes#Indexes-DuplicateValues, but I suspect the real solution is to not require uniqueness in this case.

If you want to ensure uniqueness only within the Items for a single user you might want to try the $addToSet option. See

<a href="https://www.mongodb.org/display/DOCS/Updating#Updating-%24addToSet">https://www.mongodb.org/display/DOCS/Updating#Updating-%24addToSet</a>

You can create a **unique compound sparse index** to accomplish something like what you are hoping for. It may not be the best option (client side still might be better), but it can do what you’re asking depending on specific requirements.

To do it, you’ll need to create another field on the same level as Name: Bob that is unique to each top-level record (could do FirstName + LastName + Address, we’ll call this key Identifier).

Then create an index like this:

ensureIndex({‘Identifier’:1, ‘Items.name’:1},{‘unique’:1, ‘sparse’:1})

A sparse index will ignore items that don’t have the field, so that should get around your NULL key issue. Combining your unique Identifier and Items.name as a compound unique index should ensure that you can’t have the same item name twice per person.

Although I should add that I’ve only been working with Mongo for a couple of months and my science could be off. This is not based on empirical evidence but rather observed behaviour.

**More on MongoDB Indexes**  
1. Compound Keys Indexes  
2. Sparse Indexes

**Related Page:**[**MongoDB Vs Elasticsearch Comparison**](https://mindmajix.com/mongodb-vs-elasticsearch)

**Q37) Creating a database in Mongo: can’t connect, get “connect failed”**

**Ans:**I want to create a new database in Mongo. However, I’m having trouble connecting:  
:~$ mongo  
MongoDB shell version: 1.6.5  
connecting to: test  
Tue Dec 21 18:16:25 Error: couldn’t connect to server 127.0.0.1 (anon):1154  
exception: connect failed

How can I connect to mongo in order to create a new database? Alternatively, can I create a new database from the command line?  
Slightly surprisingly, the Mongo docs don’t seem to cover how to create a database.  
Thanks.

In order to open Mongo JavaScript shell, a Listener should be initialized first.  
So, first run mongod.exe before running mongo.exe. Both are in the same location(/bin).

There is no separate commands to create a db in mongodb. Just type “use dbname;” in console. Now you have created a db of the name ‘dbname’. Now, if you type ‘show databases’ you cannot see the db name you just created. Because, mongo will not create any db, util you create collection and insert a document into that collection.  
Hope this is useful to you!

1. cd /var/lib/mongodb/  
2. Remove mongod.lock file from this folder  
3. Sudo start mongodb (in console)  
4. Mongo (in console)

And it runs fine. First you’ll need to run mongod on one terminal. Then fire up another terminal and type mongo. This shall open the mongo shell. You also need to create /data/db/ where mongo will store your databases.

Q38) How can I use MongoDB to find all documents which have a field, regardless of the value of that field?

For example, I have collection with documents, where documents can have field “url” (but most of them doesn’t). How can I find all documents, which have field “url” (regardless of value of this field)? To find if a key/field exists in your document use the $exists operator.  
Via the MongoDB shell …

> db.things.find( { url : { $exists : true } } );

**Q39) Mongodb Query To select records having a given key**

**Ans:**let the records in database are

|  |  |
| --- | --- |
| 1  2  3  4 | {“\_id”:”1?,”fn”:”sagar”,”ln”:”Varpe”}  {“\_id”:”1?,”fn”:”sag”,”score”:”10?}  {“\_id”:”1?,”ln”:”ln1?,” score”:”10?}  {“\_id”:”1?,”ln”:”ln2?} |

I need to design a MongoDB query to find all records who has a given key  
like if i pass “ln” as a parameter to query it shold return all records in which “ln”is a Key , the results fo are

|  |  |
| --- | --- |
| 1  2  3 | {“\_id”:”1?,”fn”:”sagar”,”ln”:”Varpe”}  {“\_id”:”1?,”ln”:”ln1?,”score”:”10?}  {“\_id”:”1?,”ln”:”ln2?} |

To find if a key/field exists in your document use the $exists operator.  
Via the MongoDB shell …

db.things.find( { ln : { $exists : true } } );

Possible duplicate of this question: MongoDB queries

**Q40) “Field name duplication not allowed with modifiers” on update**

**Ans:**I get a “Field name duplication not allowed with modifiers” error while trying to update a field(s) in Mongo. An example:

|  |  |
| --- | --- |
| 1  2  3  4  5  6 | > db.test.insert({test: “test1”, array: [0]});  > var testFetch = db.test.findOne({test: “test1”});  > db.test.update(testFetch,  {$push: {array: 1}, //push element to end of key “array”  $pop:  {array: -1} //pop element from the start of key “array”  }); |

Field name duplication not allowed with modifiers  
Is there no way to perform this atomic operation? I don’t want to be doing two separate updates for this.  
There’s an outstanding issue for this on Mongo’s ticket system: https://jira.mongodb.org/browse/SERVER-1050  
Looks like it’s scheduled for this year. Your scenario is definitely a sensible scenario, but it’s also tied to a bunch of edge cases. What if you $push and $pop on an empty array? What’s expected? What do you want if you $push and $pull?  
I don’t want to be doing two separate updates for this.  
I know that doing this really has “code smell”, but is it a complete blocker for using this solution? Is the “double-update” going to completely destroy server performance?

**Q41) In mongoDb, how do you remove an array element by its index**

**Ans:**

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11 | Skip code block  {  “\_id” : ObjectId(“4d1cb5de451600000000497a”),  “name” : “dannie”,  “interests” : [  “guitar”,  “programming”,  “gadgets”,  “reading”  ]  } |

In the example above, assume the above document is in the **db.people**collection. How to remove the 3rd element of the **interests**array by it’s **index**?

Edit:  
This is my current solution:

|  |  |
| --- | --- |
| 1  2  3 | var interests = db.people.findOne({“name”:”dannie”}).interests;  interests.splice(2,1)  db.people.update({“name”:”dannie”}, {“$set” : {“interests” : interests}}); |

Is there a more direct way?

There is no straight way of pulling/removing by array index. In fact, this is an opened issue

<a href="https://jira.mongodb.org/browse/SERVER-1014,"> https://jira.mongodb.org/browse/SERVER-1014,</a>

you may vote for it.

|  |  |
| --- | --- |
| 1  2  3 | The workaround is using $unset and then $pull:  db.lists.update({}, {$unset : {“interests.3” : 1 }})  db.lists.update({}, {$pull : {“interests” : null}}) |

Update: as mentioned in some of the comments this approach is not atomic and can cause some race conditions if other clients read and/or write between the two operations. If we need the operation to be atomic, we could:

1. Read the document from the database  
2. Update the document and remove the item in the array  
3. Replace the document in the database. To ensure the document has not changed since we read it, we can use the update if current pattern describedin the mongo docs.

You can use $pull modifier of update operation for removing a particular element in an array. In case you provided a query will look like this:  
db.people.update({“name”:”dannie”}, {‘$pull’: {“interests”: “guitar”}})

Also, you may consider using $pullAll for removing all occurrences. More about this on the official documentation page –

<a href="https://www.mongodb.org/display/DOCS/Updating#Updating-%24pull">https://www.mongodb.org/display/DOCS/Updating#Updating-%24pull</a>

This doesn’t use index as a criteria for removing an element, but still might help in cases similar to yours. IMO, using indexes for addressing elements inside an array is not very reliable since mongodb isn’t consistent on an elements order as fas as I know.

**Q42) MongoDB custom and unique IDs**

**Ans:**

I’m using MongoDB, and I would like to generate unique and cryptical IDs for blog posts (that will be used in restful URLS) such as s52ruf6wst or xR2ru286zjI.

What do you think is best and the more scalable way to generate these IDs ?  
I was thinking of following architecture :

1. A periodic (daily?) batch running to generate a lot of random and uniques IDs and insert them in a dedicated MongoDB collection with InsertIfNotPresent  
2. And each time I want to generate a new blog post, I take an ID from this collection and mark it as “taken” with UpdateIfCurrent atomic operation  
WDYT? This is exactly why the developers of MongoDB constructed their ObjectID’s (the \_id) the way they did … to scale across nodes, etc.

A BSON ObjectID is a 12-byte value consisting of a 4-byte timestamp (seconds since epoch), a 3-byte machine id, a 2-byte process id, and a 3-byte counter. Note that the timestamp and counter fields must be stored big endian unlike the rest of BSON. This is because they are compared byte-by-byte and we want to ensure a mostly increasing order. Here’s the schema:

0123   456      78    91011

time   machine  pid   inc  
Traditional databases often use monotonically increasing sequence numbers for primary keys. In MongoDB, the preferred approach is to use Object IDs instead. Object IDs are more synergistic with sharding and distribution.

<a href="https://www.mongodb.org/display/DOCS/Object+IDs">https://www.mongodb.org/display/DOCS/Object+IDs</a>

So I’d say just use the ObjectID’s  
They are not that bad when converted to a string (these were inserted right after each other) …  
For example:

4d128b6ea794fc13a8000001  
4d128e88a794fc13a8000002

They look at first glance to be “guessable” but they really aren’t that easy to guess …

4d128 b6e a794fc13a8000001  
4d128 e88 a794fc13a8000002

And for a blog, I don’t think it’s that big of a deal … we use it production all over the place.  
What about using UUIDs?

https://www.famkruithof.net/uuid/uuidgen as an example.

**Q43) MongoDB shell run() function**

**Ans:**run(“/test.js”)

in mongo shell. i always get error

run(“/test.js”) shell: started program /test.js sh2516| Unable to start program /test.js errno:13 Permission denied 255

for test i add 777 permission on this file i doesn’t have any passwords on DB  
test.js is empty

What am I doing wrong?  
try using load() rather than run(). load is for javascript while run is for system binaries.

**Q44) “system.” In a Collection Name in MongoDB**

**Ans:**I just discovered a bizarre behavior exhibited by MongoDB.  
Apparently, any collection name with the string “system.” anywhere in it will just not function correctly.  
To make matters worse, it won’t even tell you anything is wrong!

It’s really more a matter of curiosity, but does anybody have any idea why this would happen? Is it documented somewhere?  
My assumption is that it uses “”system.\*” collections to store things internally (like indexes) and doesn’t want you messing with them, but this doesn’t seem like the correct behavior to me.  
You are correct “system.\*” is a reserved collection namespace used by MongoDB in each DB.  
It is used to store indexes and users, etc.

SQL Server has many such tables too, and I don’t believe they warn you not to use them either :)  
But you could always put in a request for such functionality:

<a href="https://jira.mongodb.org/">https://jira.mongodb.org/</a>

You can see them by running …  
> show collections  
and you’ll see something like …  
system.indexes  
system.users  
So, you can see your indexes for example:  
> db.system.indexes.find()  
From the MongoDB docs:

The **.system.**\* namespaces in MongoDB are special and contain database system information. System collections include:  
1. **system.namespaces**lists all namespaces.  
2. **system.indexes** lists all indexes.  
3. Additional namespace / index metadata exists in the database.ns files, and is opaque.  
4. **system.profile**stores database profiling information.  
5.**system.users**lists users who may access the database.  
6. **local.sources**stores replica slave configuration data and state.  
7. Information on the structure of a stored object is stored within the object itself. See BSON .

There are several restrictions on manipulation of objects in the system collections. Inserting in system.indexes adds an index, but otherwise that table is immutable (the special drop index command updates it for you). **system.users** is modifiable. **system.profile** is droppable.

https://docs.mongodb.org/manual/reference/system-collections/

**Q45) MongoDB Shell – access collection with period in name?**

**Ans:**I have found a collection in one of our MongoDB databases with the name my.collection.  
Is there a way to access this collection from the MongoDB shell, despite it having a point in the name?  
> db.my.collection.findOne();  
null  
I’m pretty sure that that is not correct.  
try this instead:  
db[“my.collection”].findOne();

you run into the same issue with hyphens or any other name that does not match on [a-zA-Z\_$][0-9a-zA-Z\_$]  
This limitation comes from valid named for javascript object properties.

if collection name is “my.collection”  
db.my.collection.findOne(); // OK  
null  
if collection name is “my.1.collection”  
db.my.1.collection.findOne(); // Not OK  
SyntaxError: missing ; before statement  
Fix:  
db[“my.1.collection”].findOne(); // Now is OK  
null

**Q46) How do I insert a record from one mongo database into another?**

**Ans:**As far as I see all commands operate on the same database in mongodb. I want to do something like this:  
db.mySourceCollection.find().forEach( function(x){ db.theDestinationCollection.save(x)} );  
where mySourceCollection is on liveDatabase and theDestinationCollection is ontestDatabase.  
Use use :-)

> var documents = db.mySourceCollection.find()<br>> use testDatabase

switched to db testDatabase

> documents.forEach(function(x){ db.theDestinationCollection.insert(x) })

db is used to refer to the currently connected database, however you can switch databases on the fly using the use command, as I’ve shown above.  
Check out the help command in the shell — it mentions this command and much more!

use dbname doesn’t work in scripted mode (i.e. when scripting the shell with javascript), so you should use the db.getSiblingDB() method instead to reassign the ‘db’ variable, e.g.:  
db = db.getSiblingDB(“otherdb”)

More info here: https://www.mongodb.org/display/DOCS/Scripting+the+shell

**Related Page:**[**MongoDB Vs. CouchDB**](https://mindmajix.com/mongodb-vs-couchdb)

**Q47) find inside a hash mongodb**

**Ans:**I have this struct in my collection:  
{foo : 1, bar : 4, baz : {a : 1, b : 2 ,c : “fafofu”}}  
How do I find “a” and “b” inside baz ? It does not works db.my\_collection.find({baz : {a : 1, b : 2});  
I don’t care about if “c” is “fafofu” or “cacocu” does not matters.  
You can use . to reach into the baz object.  
db.my\_collection.find({“baz.a” : 1, “baz.b” : 2});

**Q48) Query IDE for MongoDB?**

**Ans:**I’m wondering if there is an IDE for MongoDB that allows you to run queries and see the results? This would behave like query analyzer in SQL Server Management Studio. The issue I’m having right now is that I have to do queries, such as “db.MyTable.find()” from command prompt, which isn’t a good solution.  
If the answer is no, is there a more mature “no sql” solution like MongoDB that does have an IDE?

**Web Based**

For PHP, I’d recommend Rock Mongo. Solid, lots of great features, easy setup.

<a href="https://code.google.com/p/rock-php/wiki/rock\_mongo">https://code.google.com/p/rock-php/wiki/rock\_mongo</a>

If you don’t want to install anything … you can use MongoHQ’s web interface (even if you your MongoDB isn’t on MongoHQ.)

<a href="https://mongohq.com/home">https://mongohq.com/home</a>

**Windows**

By far, the best UI (for Windows) currently out there is MongoVUE.

<a href="https://blog.mongovue.com/">https://blog.mongovue.com/</a>

Looks great, lots of features, and if you are new it will really help you get going

<a href="https://blog.mongovue.com/features/">https://blog.mongovue.com/features/</a>

Here’s a Q&A with the author too if you are interested

<a href="https://learnmongo.com/posts/qa-ishann-kumar-creator-of-mongovue/">https://learnmongo.com/posts/qa-ishann-kumar-creator-of-mongovue/</a>

There is an official list of admin tools here:

<a href="https://www.mongodb.org/display/DOCS/Admin+UIs">https://www.mongodb.org/display/DOCS/Admin+UIs</a>

Another contender : https://www.robomongo.org/ Robomongo give you a shell like interface but outputs your results in the gui. Its available for windows, mac(dmg, zip) and linux (deb, rpm, tar.gz) as a desktop application. Currently its free!

Robomongo prints the results in a treeView or Json text representation and supports the generation of UUID (.NET-,Python-,Java-Encoding). It has autocomplete, shows multiple results at once and has a query history.

**Q49) MongoDB: Geting “Client Cursor::yield can’t unlock b/c of recursive lock” warning when use findAndModify in two process instances**

**Ans:**I’m using: MongoDB 1.6.4, Python 2.6.6, PyMongo 1.9, Ubuntu 10.10  
I’m getting “Client Cursor::yield can’t unlock b/c of recursive lock” warning in my logs very often when use findAndModify in two process instances. When i use only one process warning doesn’t appear.  
How can i fix this?  
\*Update 8 March 2013 \*  
Is there a fix to this problem as of now?  
this is usually means you are missing indexes on fields used in query.  
I don’t know tech details of this warning but from my experience adding index on the query field helps. check you have index on fields that used in query part of findAndModify. also run db.collection.find().explain() to check if it uses the index.  
Thanks to the pingw33n who help solve this question.

**Q50) how to query child objects in mongodb**

**Ans:**I’m new to mongodb and am trying to query child objects. I have a collection of States, and each State has child Cities. One of the Cities has a Name property that is null, which is causing errors in my app. How would I query the State collections to find child Cities that have a name == null?

If it is exactly null (as opposed to not set):  
db.states.find({“cities.name”: null})

(but as javierfp points out, it also matches documents that have no cities array at all, I’m assuming that they do).

If it’s the case that the property is not set:  
db.states.find({“cities.name”: {“$exists”: false}})

I’ve tested the above with a collection created with these two inserts:

]db.states.insert({“cities”: [{name: “New York”}, {name: null}]})  
db.states.insert({“cities”: [{name: “Austin”}, {color: “blue”}]})

The first query finds the first state, the second query findsthe second. If you want to find them both with one query you can make an $or query:

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9 | db.states.find({“$or”: [  {“cities.name”: null},  {“cities.name”: {“$exists”: false}}  ]})  Assuming your “states” collection is like:  {“name” : “Spain”, “cities” : [ { “name” : “Madrid” }, { “name” : null } ] }  {“name” : “France” }  The query to find states with null cities would be:  db.states.find({“cities.name” : {“$in” : [null], “$exists” : true}}); |

It is a common **mistake**to query for nulls as:  
db.states.find({“cities.name” : null});  
because this query will return all documents lacking the key (in our example it will return Spain and France). So, unless you are sure the key is always present you must check that the key exists as in the first qeury.

**Q51) MongoDB 1.6.5: how to rename field in collection**

**Ans:**$rename function is available only in development version 1.7.2. How to rename field in 1.6.5?  
The simplest way to perform such an operation is to loop through the data set re-mapping the name of the field. The easiest way to do this is to write a function that performs the re-write and then use the .find().forEach() syntax in the shell.  
Here’s a sample from the shell:

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10 | Skip code block  db.foo.save({ a : 1, b : 2, c : 3});  db.foo.save({ a : 4, b : 5, c : 6});  db.foo.save({ a : 7, b : 8 });  db.foo.find();  remap = function (x) {  if (x.c){  db.foo.update({\_id:x.\_id}, {$set:{d:x.c}, $unset:{c:1}});  }  } |
| 1  2 | db.foo.find().forEach(remap);  db.foo.find(); |

In the case above I’m doing an $unset and a $set in the same action. MongoDB does not support transactions across collections, but the above is a single document. So you’re guaranteed that the set and unset will be atomic (i.e. they both succeed or they both fail).

The only limitation here is that you’ll need to manage outside writers to keep the data consistent. My normal preference for this is simply to turn off writes while this updates. If this option is not available, then you’ll have to figure out what level of consistency you want for the data. (I can provide some ideas here, but it’s really going to be specific to your data and system)  
db.collection\_name.update({}, {$rename: {“oldname”: “newname”}}, false, true);

This will rename the column for each row in the collection.  
Also, I discovered that if your column (the one you’re renaming) appears within the index catalog (db.collection\_name.getIndexes()), then you’re going to have to drop and recreate the index (using the new column name) also.

**Q52) Is it normal size for MongoDB?**

**Ans:**I just import my **Mysql**database (size 2,8 Mio) in my new **Mongo**database with very simple php script i build, import was ok without error but when i look my **Mongo**database (with RockMongo) i can see this : Data Size 8.01m, Storage Size 13.7m.

**MongoDB**is bigger than **Mysql**for the same amount of data, is this normal?  
Thanks for your help and sorry for my english.

Yes, it’s normal that the “same” data will take up more space in mongodb. There’s a few things you need to take into account:  
1) the document \_id that’s stored for each document (unless you are specifying your own value for that) is 12 bytes per doc  
2) you’re storing the key for each key-value pair in each document, whereas in MySql the column name is not stored for every single row so you have that extra overhead in your mongodb documents too. One way to reduce this is to use shortened key names (“column names”) in your docs  
3) mongodb automatically adds padding to allow documents to grow  
In similar tests, loading data from SQL Server to MongoDB, with shortened 2 character document key names instead of the full names as per SQL Server, I see about 25-30% extra space being used in MongoDB.

**Q53) Maximum number of databases supported by MongoDB**

**Ans:**I would like to create a database for each customer. But before, I would like to know how many databases can be created in a single instance of MongoDB ?  
There’s no explicit limit, but there are probably some implicit limits due to max number of open file handles / files in a directory on the host OS/filesystem.  
see: https://groups.google.com/group/mongodb-user/browse\_thread/thread/01727e1af681985a?fwc=2

**Q54) Mongo $in operator performance**

**Ans:**Is it slow / poor form to use the $in operator in MongoDB with a large array of possibilities?

|  |  |
| --- | --- |
| 1  2  3  4  5 | posts.find({  author : {  $in : [‘friend1',’friend2',’friend3’….’friend40’]  }  }) |

App Engine, for example, won’t let you use more than 30 because they translate directly to one query per item in the IN array, and so instead force you into using their method for handling fan out. While thats probably the most efficient method in Mongo too, the code for it is significantly more complex so I’d prefer to just use this generic method.  
Will Mongo execute these $in queries efficiently for reasonable-sized datasets?

It can be fairly efficient with small lists (hard to say what small is, but at least into the tens/hundreds) for $in. It does not work like app-engine since mongodb has actual btree indexes and isn’t a column store like bigtable.

With $in it will skip around in the index to find the matching documents, or walk through the whole collection if there isn’t an index to use.  
If you build an index (ensureIndex) on the list element, it should be pretty quick.

Have you tried using explain()? Its a good, built-in way to profile your queries: https://www.mongodb.org/display/DOCS/Indexing+Advice+and+FAQ#IndexingAdviceandFAQ-Use%7B%7Bexplain%7D%7D.

**Q55) Mongo Query question $gt,$lt**

**Ans:**I have a query below. I want get items between 4 and 6 so only a:1 should match because it has the value 5 in b.

|  |  |
| --- | --- |
| 1  2  3  4 | > db.test.find({ b : { $gt :  4  }, b: {$lt : 6}});  { “\_id” : ObjectId(“4d54cff54364000000004331”), “a” : 1, “b” : [ 2, 3, 4, 5 ] }  { “\_id” : ObjectId(“4d54d0074364000000004332”), “a” : 2, “b” : [ 2, 4, 6, 8 ] }  > |

Can someone tell be why a:2 is matching this query? I can’t really see why it is being returned.  
I also tried what was specified in the tutorial but id did not seem to work:

|  |  |
| --- | --- |
| 1  2  3  4 | > db.test.find({ b : { $gt :  4, $lt : 6}});  { “\_id” : ObjectId(“4d54cff54364000000004331”), “a” : 1, “b” : [ 2, 3, 4, 5 ] }  { “\_id” : ObjectId(“4d54d0074364000000004332”), “a” : 2, “b” : [ 2, 4, 6, 8 ] }  > |

And this one to avoid any confusion regarding GT/GTE

|  |  |
| --- | --- |
| 1  2  3  4 | > db.test.find({b: {$gt: 4.5, $lt: 5.5}});  { “\_id” : ObjectId(“4d54cff54364000000004331”), “a” : 1, “b” : [ 2, 3, 4, 5 ] }  { “\_id” : ObjectId(“4d54d0074364000000004332”), “a” : 2, “b” : [ 2, 4, 6, 8 ] }  > |

only a:1 should be returned.  
As suggested, I gave $elemMatch a try but it did not appear to work either (objectIds are different because I am on a different machine)

|  |  |
| --- | --- |
| 1  2  3  4  5 | > db.test.find();  { “\_id” : ObjectId(“4d5a24a5e82e00000000433f”), “a” : 1, “b” : [ 2, 3, 4, 5 ] }  { “\_id” : ObjectId(“4d5a24bbe82e000000004340”), “a” : 2, “b” : [ 2, 4, 6, 8 ] }  > db.test.find({b: {$elemMatch: {$gt : 4, $lt: 6 }}});  > |

No documents were returned.  
This is a really confusing topic. I work at 10gen and I had to spend a while wrapping my head around it  
Let’s walk through how the query engine processes this query.  
Here’s the query again:

> db.test.find({ b : { $gt :  4, $lt : 6}});

When it gets to the record that seems like it shouldn’t match…

{ “\_id” : ObjectId(“4d54cff54364000000004331”), “a” : 1, “b” : [ 2, 4, 6, 8 ] }

The match is not performed against each element of the array, but rather against the array as a whole.  
The comparison is performed in three steps:

**Step 1:** Find all documents where b has a value greater than 4  
b: [2,4,6,8] matches because 6 & 8 are greater than 4

**Step 2**: Find all documents where b has a value less than 6  
b: [2,4,6,8] matches because 2 & 4 are less than 6

**Step 3:** Find the set of documents that matched in both step 1 & 2.  
The document with b: [2,4,6,8] matched both steps 1 & 2 so it is returned as a match. Note that results are also de-duplicated in this step, so the same document won’t be returned twice.

If you want your query to apply to the individual elements of the array, rather than the array as a whole, you can use the $elemMatch operator. For example

|  |  |
| --- | --- |
| 1  2  3 | > db.temp.find({b: {$elemMatch: {$gt: 4, $lt: 5}}})  > db.temp.find({b: {$elemMatch: {$gte: 4, $lt: 5}}})  { “\_id” : ObjectId(“4d558b6f4f0b1e2141b66660”), “b” : [ 2, 3, 4, 5, 6 ] } |

**Related Page:**[**MongoDB Show Collections**](https://mindmajix.com/mongodb-show-collections)

**Q56) MongoDB: How to change the type of a field?**

**Ans:**There is a question already in Stackoverflow, very similar with my question. The thing is that the answer for that questions was for a Java Driver, I am trying to do it in the shell.  
I am doing this…

db.meta.update({‘fields.properties.default’: { $type : 1 }}, {‘fields.properties.default’: { $type : 2 }})

This is not working  
The only way to change the $type of the data is to perform an update on the data where the data has the correct type.

In this case, it looks like you’re trying to change the $type from 1 (double) to 2 (string).

https://www.mongodb.org/display/DOCS/Advanced+Queries#AdvancedQueries-%24type

So simply load the document from the DB, perform the cast (new String(x)) and then save the document again.

If you need to do this programmatically and entirely from the shell, you can use thefind(…).forEach(function(x) {}) syntax.  
In response to the second comment below. Change the field bad from a number to a string in collection foo.

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16 | db.foo.find( { ‘bad’ : { $type : 1 } } ).forEach( function (x) {  x.bad = new String(x.bad); // convert field to string  db.foo.save(x);  });  // String to Integer  db.db-name.find({field-name : {$exists : true}}).forEach( function(obj) { obj.field-name = new NumberInt( obj.field-name ); db.db-name.save(obj); } );  // Integer to String  db.db-name.find({field-name : {$exists : true}}).forEach( function(obj) { obj.field-name = “”+obj.field-name; db.db-name.save(obj); } );  This is what I used for string to int conversion.  db.my\_collection.find().forEach( function(obj) {  obj.my\_value= parseInt(obj.my\_value);  db.my\_collection.save(obj);  });  db.coll.find().forEach(function(data) {  db.coll.update({\_id:data.\_id},{$set:{myfield:parseInt(data.myfield)}});  }) |

**Q57) Many to many update in MongoDB without transactions**

**Ans:**I have two collections with a many-to-many relationship. I want to store an array of linked ObjectIds in both documents so that I can take Document A and retrieve all linked Document B’s quickly, and vice versa.

Creating this link is a two step process  
1. Add Document A’s ObjectId to Document B  
2. Add Document B’s ObjectId to Document A

After watching a MongoDB video I found this to be the recommended way of storing a many-to-many relationship between two collections  
I need to be sure that both updates are made. What is the recommended way of robustly dealing with this crucial two step process without a transaction?  
I could condense this relationship into a single link collection, the advantage being a single update with no chance of Document B missing the link to Document A. The disadvantage being that I’m not really using MongoDB as intended. But, because there is only a single update, it seems more robust to have a link collection that defines the many-to-many relationship.

Should I use safe mode and manually check the data went in afterwards and try again on failure? Or should I represent the many-to-many relationship in just one of the collections and rely on an index to make sure I can still quickly get the linked documents?  
Any recommendations? Thanks

**@Gareth**, you have multiple legitimate ways to do this. So they key concern is how you plan to query for the data, (i.e.: what queries need to be fast)  
Here are a couple of methods.

**Method #1: the “links” collection**  
You could build a collection that simply contains mappings between the collections.  
Pros:  
1. Supports atomic updates so that data is not lost  
Cons:  
2. Extra query when trying to move between collections

**Method #2: store copies of smaller mappings in larger collection**  
For example: you have millions of Products, but only a hundred Categories. Then you would store the Categories as an array inside each Product.  
Pros:  
1. Smallest footprint  
2. Only need one update  
Cons:  
1. Extra query if you go the “wrong way”

**Method #3: store copies of all mappings in both collections**  
(what you’re suggesting)  
Pros:  
1. Single query access to move between either collection  
Cons:  
1. Potentially large indexes  
2. Needs transactions (?)  
Let’s talk about “needs transactions”. There are several ways to do transactions and it really depends on what type of safety you require.  
Should I use safe mode and manually check the data went in afterwards and try again on failure?  
You can definitely do this. You’ll have to ask yourself, what’s the worst that happens if only one of the saves fails?

**Method #4: queue the change**

I don’t know if you’ve ever worked with queues, but if you have some leeway you can build a simple queue and have different jobs that update their respective collections.  
This is a much more advanced solution. I would tend to go with #2 or #3.

**Q58) MongoDB log file growth**

**Ans:**Currently my log file sits at 32 meg. Did I miss an option that would split the log file as it grows?  
Rotate the logs yourself

<a href="https://www.mongodb.org/display/DOCS/Logging">https://www.mongodb.org/display/DOCS/Logging</a>

or use ‘logrotate’ with an appropriate configuration.  
You can use logrotate to do this job for you.  
Put this in /etc/logrotate.d/mongod (assuming you use Linux and have logrotated installed):  
Skip code block

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13 | /var/log/mongo/\*.log {  daily  rotate 30  compress  dateext  missingok  notifempty  sharedscripts  copytruncate  postrotate  /bin/kill -SIGUSR1 `cat /var/lib/mongo/mongod.lock 2> /dev/null` 2> /dev/null || true  endscript  } |

If you think that 32 megs is too large for a log file, you may also want to look inside to what it contains.  
If the logs seem mostly harmless (“open connection”, “close connection”), then you may want to start mongod with the –quiet switch. This will reduce some of the more verbose logging.

**Q59) MongoDB field order and document position change after update**

**Ans:**I am learning MongoDB and I noticed that whenever I do an update on a document the field being updated is pushed to the end of the order, so if I had something like:

|  |  |
| --- | --- |
| 1  2  3  4  5  6 | db.collection.save({field1: value1, field2: value2, …, field 10: value10});  db.collection.update({field1: value1}, {$set: {field2: new\_value}});  then if you do:  db.collection.find();  it will display:  { “field1?:”value1”, …, “field10?:”value10”, “field2?:”new\_value”} |

You can see how the field order changes where the updated field is being pushed to the end of the document. In addition, the document itself is being pushed to the end of the collectoin. I know that it’s a “schema-less” DB and it may not be a huge problem, but it just doesn’t look “pretty” :). Is there a way to do an in-place update without changing the order?  
MongoDB allocates space for a new document based on a certain padding factor. If your update increases the size of the document beyond the size originally allocated the document will be moved to the end of the collection. The same concept applies to fields in a document.

Both document structure and collection structure in MongoDB based on JSON principles. JSON is a **set**of key/value pairs (in particular fieldName/fieldValue for document and index/document for collection). From this point of view it doesn’t seem that you can relay on order at all.

In the case of the documents if the field size changes, it writes out a new document with the fields sorted by field name. This behavior can be seen with the following statements  
Case 1: No change in size of field, so no change in field order

db.testcol.find() db.testcol.save({a:1,c:3,b:2}) db.testcol.find() { “\_id” : ObjectId(“4d5efc3bec5855af36834f5a”), “a” : 1, “c” : 3, “b” : 2 } db.testcol.update({a:1},{$set:{c:22}}) db.testcol.find() { “\_id” : ObjectId(“4d5efc3bec5855af36834f5a”), “a” : 1, “c” : 22, “b” : 2 }

Case 2: Field size changes and the fields are reodered

db.testcol.find() db.testcol.save({a:1,c:”foo”,b:2,d:4}) db.testcol.find() { “\_id” : ObjectId(“4d5efdceec5855af36834f5e”), “a” : 1, “c” : “foo”, “b” : 2, “d” : 4 } db.testcol.update({a:1},{$set:{c:”foobar”}}) db.testcol.find() { “\_id” : ObjectId(“4d5efdceec5855af36834f5e”), “a” : 1, “b” : 2, “c” : “foobar”, “d” : 4 }

Is there a particular reason why you do not want the fields reordered? The above was using 1.8.0\_rc0 on OS X  
FYI, in MongoDB 2.6 updates will preserve field order, with the following exceptions:  
1. The \_id field is always the first field in the document.  
2. Updates that include renaming of field names may result in the reordering of fields in the document.

**Related Page:**[**MongoDB Find Queries**](https://mindmajix.com/mongodb-find-queries)

**Q60) Limit MongoDB database size?**

**Ans:**We’re interested in deploying MongoDB, but we need to know if we can limit database/table sizes?

For example:  
db.user1.find() db.user2.find()  
As you can see from the above, each user will have their own database. We want to limit each user’s database so we don’t have any one user eating up all our hard drive space.

Is this possible with MongoDB?  
Thanks.

You can create a database per user and enable the –quota option. This will allow you keep any user from using too much space.

https://www.mongodb.org/display/DOCS/Command+Line+Parameters<br>https://www.mongodb.org/display/DOCS/Excessive+Disk+Space

**Q61) While Creating New User in MongoDB, it has to be created under admin?**

**Subscribe to our youtube channel to get new updates..!**

**Ans:**I want to create a new user in MongoDB, So i have to do that by login to admin by(use admin) and then i have to use the command add user to create new user is it?  
Thanks,  
If no users created you can create new user without any authentification, but if you have created admin user for specific database you should authentifcate, and then perform any operation.

**Documentation:**  
If no users are configured in admin.system.users, one may access the database from the localhost interface without authenticating. Thus, from the server running the database (and thus on localhost), run the database shell and configure an administrative user:

|  |  |
| --- | --- |
| 1  2  3 | $ ./mongo  > use admin  > db.addUser(“theadmin”, “anadminpassword”) |

We now have a user created for database admin. Note that if we have not previously authenticated, we now must if we wish to perform further operations, as there is a user in admin.system.users.

> db.auth(“theadmin”, “anadminpassword”)

We can view existing users for the database with the command:

> db.system.users.find()

Now, let’s configure a “regular” user for another database.

> use projectx<br>> db.addUser(“joe”, “passwordForJoe”)

**Q62) When to embed documents in Mongo DB**

**Ans:**I’m trying to figure out how to best design Mongo DB schemas. The Mongo DB documentation recommends relying heavily on embedded documents for improved querying, but I’m wondering if my use case actually justifies referenced documents.  
A very basic version of my current schema is basically: (Apologies for the psuedo-format, I’m not sure how to express Mongo schemas)  
Skip code block

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13 | users {  email (string)  }  games {  user (reference user document)  date\_started (timestamp)  date\_finished (timestamp)  mode (string)  score: {  total\_points (integer)  time\_elapsed (integer)  }  } |

Games are short (about 60 seconds long) and I expect a lot of concurrent writes to be taking place.

At some point, I’m going to want to calculate a high score list, and possibly in a segregated fashion (e.g., high score list for a particular game.mode or date)  
Is embedded documents the best approach here? Or is this truly a problem that relations solves better? How would these use cases best be solved in Mongo DB?  
… is this truly a problem that relations solves better?

The key here is less about “is this a relation?” and more about “how am I going to access this?”

MongoDB is not “anti-reference”. MongoDB does **not**have the benefits of joins, but it **does**have the benefit of embedded documents.  
As long as you understand these trade-offs then it’s perfectly fair to use references in MongoDB. It’s really about how you plan to query these objects.  
Is embedded documents the best approach here?

Maybe. Some things to consider.  
1. Dogames have value outside of the context of the user?  
2. How manygames will a single user have?  
3. Isgames transactional in nature?  
4. How are you going to accessgames? Do you always need all of a user’s games?

If you’re planning to build leaderboards and a user can generate hundreds of game documents, then it’s probably fair to have games in their own collection. Storing ten thousand instances of “game” inside of each users isn’t particularly useful.

But depending on your answers to the above, you could really go either way. As the litmus test, I would try running some Map / Reduce jobs (i.e. build a simple leaderboard) to see how you feel about the structure of your data.

**Q63) Why Is MongoDB So Fast**

**Ans:**I was showing my co-worker performance benchmarks of MongoDB vs SQL 2008 and while he believes MongoDB is faster, he doesn’t understand how its possible. His logic, was that SQL has been around for decades, and has some of the smartest people working on it, and how can MongoDB; a relatively new kid on the block be so superior in performance? I wasn’t able to really provide a solid and technical answer, and I was hoping you guys could assist.

MongoDB isn’t like a traditional relational database. It’s noSQL or document based, it provides weak consistency guarantees, and it doesn’t have to guarantee consistency like SQL.  
MongoDB is fast because its web scale!

Its a fun video and well worth everyone watching, but it does answer your question – that most of the noSQL engines like MongoDB are not robust and not resilient to crashes and other outages. This security is what they sacrifice to gain speed.

SQL has to do quite a lot, Mongo just has to drop bits onto disk (almost)

**Q64) How to remove an element from a doubly-nested array in a MongoDB document**

**Ans:**I have a document structure something along the lines of the following:

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17 | {  “\_id” : “777”,  “someKey” : “someValue”,  “someArray” : [  {  “name” : “name1”,  “someNestedArray” : [  {  “name” : “value”  },  {  “name” : “delete me”  }  ]  }  ]  } |

I want to delete the nested array element with the value “delete me”.

I know I can find documents which match this description using nested $elemMatch expressions. What is the query syntax for removing the element in question?  
To delete the item in question you’re actually going to use an update. More specifically you’re going to do an update with the $pull command which will remove the item from the array.

db.temp.update({ \_id : “777” }, {$pull : { “someArray.0.someNestedArray” : {“name”:”delete me”} } } )

There’s a little bit of “magic” happening here. Using .0 indicates that we know that we are modifying the 0th item of someArray. Using {“name”:”delete me”} indicates that we know the exact data that we plan to remove.

This process works just fine if you load the data into a client and then perform the update. This process works less well if you want to do “generic” queries that perform these operations.  
I think it’s easiest to simply recognize that updating arrays of sub-documents generally requires that you have the original in memory at some point.

In response to the first comment below, you can probably help your situation by changing the data structure a little

Skip code block

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12 | “someObjects” : {  “name1”:  {  “someNestedArray” : [  {  “name” : “value”  },  {  “name” : “delete me”  }  ]  }  } |

Now you can do {$pull : { “someObjects.name1.someNestedArray” : …  
Here’s the problem with your structure. MongoDB does not have very good support for manipulating “sub-arrays”. Your structure has an array of objects and those objects contain arrays of more objects.

If you have the following structure, you are going to have a difficult time using things like $pull:

|  |  |
| --- | --- |
| 1  2  3  4 | array [  { subarray : array [] },  { subarray : array [] },  ] |

If your structure looks like that and you want to update subarray you have two options: 1. Change your structure so that you can leverage $pull. 2. Don’t use $pull. Load the entire object into a client and use findAndModify.

**Related Page:**[**MongoDB Port**](https://mindmajix.com/mongodb-port)

**Q65) Can MongoDB work when size of database larger then RAM? or when index larger then RAM?**

**Ans:**Yes it can work. To what level it will perform is more of an “It Depends”

The key thing is to ensure your **working set**can fit in RAM. So if you have 16GB of RAM and 20GB database (inc. indexes) for example, if you need to only access half of all the data as the other half is older/never actually queried then you’ll be fine as only half of your database needs to be in RAM (10GB).

Working set is key here. For example, if you have a logging application outputting to MongoDB, it may be that your working set is the amount of data (and indexes) from the past 3 months and that all data before that you don’t access.

When your working set exceeds the amount of RAM, then it will carry on working but with noticeably degraded performance as things will then be constantly having to go to disk which is far less performant. If you’re in this situation of exceeding RAM constraints on a machine, then this is where sharding comes into play – so you can balance the data out over a number of machines therefore increasing the amount of data that can be kept in RAM.

**Q66) Multiple $inc updates in MongoDB**

**Ans: Is it possible to update a single document by passing two $inc operators in a single update document?**

For example, I am trying to increment two different fields in a given document using the following update document:

|  |  |
| --- | --- |
| 1  2  3  4 | {  “$inc” : { “ViewAggregates.4d75b891842f2d3930cf7674” : 1 },  “$inc” : { “ViewAggregates.Total” : 1 }  } |

No errors are thrown and the document is updated but only one of the fields has been incremented. It is as if the server disregarded the first $inc operator and only the second was actually applied.

Is this the intendedcorrect behavior or is there something I am missing?  
This is an interesting side-effect of dictionary keys being unique — the second $inc overwrites the first.

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7 | However, it’s still possible to increment more than one field:  {  “$inc”: {  “ViewAggregates.4d75b891842f2d3930cf7674” : 1,  “ViewAggregates.Total” : 1  }  } |

This works for many other operators too :-)

**Q67) MongoDB : Indexes order and query order must match?**

**Ans:**This question concern the internal method to manage indexes and serching Bson Documents.  
When you create a multiple indexes like “index1”, “index2”, “index3?…the index are stored to be used during queries, but what about the order of queries and the performance resulting.  
sample

index1,index2,index3—-> query in the same order index1,index2,index3 (best case) index1,index2,index3—-> query in another order index2,index1,index3 (the order altered)

Many times you use nested queries including these 3 index and others items or more indexes. The order of the queries would implicate some time lost?. Must passing the queries respecting the indexes order defined or the internal architecture take care about this order search? I searching to know if i do take care about this or can make my queries in freedom manier.  
Thanks.

The order of the conditions in your query does not affect whether it can use an index or no.

|  |  |
| --- | --- |
| 1  2  3  4  5 | e.g. typical document structure:  {  “FieldA” : “A”,  “FieldB” : “B”  } |

If you have an compound index on A and B :

db.MyCollection.ensureIndex({FieldA : 1, FieldB : 1})

Then both of the following queries will be able to use that index:

|  |  |
| --- | --- |
| 1  2 | db.MyCollection.find({FieldA : “A”, FieldB : “B”})  db.MyCollection.find({FieldB : “B”, FieldA : “A”}) |

So the ordering of the conditions in the query do not prevent the index being used – which I think is the question you are asking.

You can easily test this out by trying the 2 queries in the shell and adding .explain() after the find. I just did this to confirm, and they both showed that the compound index was used.  
however, if you run the following query, this will NOT use the index as FieldA is not being queried on:

db.MyCollection.find({FieldB : “B”})

So it’s the ordering of the fields in the index that defines whether it can be used by a query and not the ordering of the fields in the query itself (this was what Lucas was referring to).

From <a href="https://www.mongodb.org/display/DOCS/Indexes:">https://www.mongodb.org/display/DOCS/Indexes:</a>

If you have a compound index on multiple fields, you can use it to query on the beginning subset of fields. So if you have an index on  
a,b,c  
you can use it query on  
a  
a,b  
a,b,c  
So yes, order matters. You should clarify your question a bit if you need a more precise answer.

**Q68) MongoDB :: are MongoIds unique across collections?**

**Ans:**I was wondering: can Mongo IDs have the same value in different collections in the same database?  
Thank you,  
The uniqueness constraint for \_id is per collection, so yes – one and the same ID can occur once per Collection.  
It’s however very unlikely, if not impossible, for the same ID to be generated twice. So in order for this to happen you would have to manually insert duplicate IDs.

**Q69) like query in mongoDB**

**Ans:**I am working on mongodb . In which i Want to use like query. my collection structure is as follows.

|  |  |
| --- | --- |
| 1  2  3  4  5 | { “name” : “xy” , “age” : 34 , “location” : “sss”}  { “name” : “xyx” , “age” : 45 , “location” : “sshs”}  { “name” : “x” , “age” : 33 , “location” : “shhss”}  { “name” : “pq” , “age” : 23 , “location” : “hhh”}  { “name” : “pqr” , “age” : 12 , “location” : “sss”} |

i want to find records matching to “name” : “x”.  
so query will return all three records matching xy ,xyz,x.  
Is it possible in mongo.  
if any one knows plz reply.  
Thanks  
You can use regular expressions to do this:  
db.customers.find( { name : /^x/i } );  
You will probably want to have some indexes on the name field.  
Read more at the MongoDB Documetation site.

**Q70) How to check if an array field contains a unique value or another array in MongoDB?**

**Ans:**I am using mongodb now.  
I have blogpost collection, and blogpost has a tags filed which is an array, e.g.

|  |  |
| --- | --- |
| 1  2  3  4 | blogpost1.tags = [‘tag1’, ‘tag2’, ‘tag3’, ‘tag4’, ‘tag5’]  blogpost2.tags = [‘tag2’, ‘tag3’]  blogpost3.tags = [‘tag2’, ‘tag3’, ‘tag4’, ‘tag5’]  blogpost4.tags = [‘tag1’, ‘tag4’, ‘tag5’] |

How can I do these searchs

|  |  |
| --- | --- |
| 1  2  3 | contains ‘tag1’  contains [‘tag1',’tag2’],  contains any of [‘tag3’, ‘tag4’] |

Try this out:

|  |  |
| --- | --- |
| 1  2  3 | db.blogpost.find({ ‘tags’ : ‘tag1’}); //1  db.blogpost.find({ ‘tags’ : { $all : [ ‘tag1’, ‘tag2’ ] }}); //2  db.blogpost.find({ ‘tags’ : { $in : [ ‘tag3’, ‘tag4’ ] }}); //3 |

My experience is that for (2) the following solution is much faster than the one with “$all”:

db.blogpost.find({ $and: [ {tags: ‘tag1’} ,{tags: ‘tag2’} ] });

but to be honest, I do not not why. I would be interested in, if anyone knows.

**Q71) create secure database in mongodb**

**Ans:**I want to create the database in mongodb and thats secure.  
here ..secure mean .application has to pass username/password to connect my database in mongodb.  
javaamtho  
From Mongo Java Tutorial

MongoDB can be run in a secure mode where access to databases is controlled through name and password authentication. When run in this mode, any client application must provide a name and password before doing any operations. In the Java driver, you simply do the following with the connected mongo object :  
boolean auth = db.authenticate(myUserName, myPassword);

If the name and password are valid for the database, auth will be true. Otherwise, it will be false. You should look at the MongoDB log for further information if available.  
Most users run MongoDB without authentication in a trusted environment.

**Configuring Authentication and Security**

Authentication is stored in each database’s system.users collection. For example, on a database projectx, projectx.system.users will contain user information.  
We should first configure an administrator user for the entire db server process. This user is stored under the special admin database.

If no users are configured in admin.system.users, one may access the database from the localhost interface without authenticating. Thus, from the server running the database (and thus on localhost), run the database shell and configure an administrative user:

|  |  |
| --- | --- |
| 1  2  3 | $ ./mongo  > use admin  > db.addUser(“theadmin”, “anadminpassword”) |

We now have a user created for database admin. Note that if we have not previously authenticated, we now must if we wish to perform further operations, as there is a user in admin.system.users.

> db.auth(“theadmin”, “anadminpassword”)

We can view existing users for the database with the command:

> db.system.users.find()

Now, let’s configure a “regular” user for another database.

> use projectx

> db.addUser(“joe”, “passwordForJoe”)

Finally, let’s add a readonly user. (only supported in 1.3.2+)

> use projectx<br>> db.addUser(“guest”, “passwordForGuest”, true)

**Related Page:**[**MongoDB Create Collection**](https://mindmajix.com/mongodb-create-collection)

**Q72) How do I copy a database from one MongoDB server to another?**

**Ans:**I have two mongodbs in different server, both start with –auth. Now I want to copy a db from one server to another.

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8 | > mongo  > use admin  > db.copyDatabase(“mydb”,”mydb”,”another\_server”)  It shows:  { “errmsg” : “”, “ok” : 0 }  And:  > db.getLastError()  null |

Seems no error, but the copy is not successful. What’s the correct command to use?  
If you are using **–auth**, you’ll need to include your username/password in there…  
Also you must be on the “destination” server when you run the command.  
db.copyDatabase(, , , , );  
If all that doesn’t work, you might want to try something like creating a slave of the database you want to copy …

**Q73) Case-insensitive query on mongodb**

**Ans:**Is there a way to query for a case-insensitive value on mongo without using map/reduce?  
Suppose you have document that contains tag field and you want search on it

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16 | Tags  {  tag,  …  }  First option is use regex(but it work slow as @RestRisiko said):  db.tags.find( { “tag” : { “$regex” : “C#”, “$options” : “-i” } })  Second option is create another, lower case field( and in mongodb it best way):  Tags  {  tag,  tagLower,  ..  }  And use find as usual:  db.tags.find( { “tagLower” : “c#”}) |

It will work faster, because above code can use index for search.  
You have to normalize the data to be queried. Using a regular expression for case-insensitive search might work as well it won’t use indexes. So your only option is to normalize. If you need to preserve the original state then you need to denormalize the data and store the normalized values in a dedicated column of the document.

**Q74) MongoDB load balancing**

**Ans:**I was looking at best load balancing option for concurrent users with Mongo DB. I have looked at Master Slave replication but don’t think this will load balance. Are there any open source DB load balancers for Mongo DB? I have looked at Sequoia but looks like that project is no longer actively supported.

Please note: The data is not very huge & also not use case for sharding. both Master Slave and Replica Sets will load balance in MongoDB, if you set slaveOK in your driver.  
When slaveOK is enabled MongoDB drivers direct all reads to secondaries/slaves. This provides relatively effective read balancing; for write balancing your only option.would be sharding.

**Q75) MongoDB: unconditional updates?**

**Ans:**This seems like a silly question but I haven’t yet found the answer. If I simply wanted to add the same field->value to EVERY record in a MongoDB collection, what would be the appropriate shell command to do so? I tried doing a multi update with a blank query ({}) but that resulted in this error:

**multi update only works with $ operators**

I’m a bit puzzled about how to get around this. Any suggestions?  
The error says it all: You can only modify multiple documents using the $ modifier operators. You probably had something like this:

> db.coll.update({ }, { a: ‘b’ }, false, true);

Which would normally replace the first object in the collection with { a: ‘b’ } if multi was false. You wouldn’t want to replace all the objects in your collection with the same document!  
Use the $set operator instead:

> db.coll.update({ }, { ‘$set’: { a: ‘b’ } }, false, true);

This will set the a property of every document (creating it as necessary) to ‘b’.

**Q76) Mongo: find items that don’t have a certain field**

**Ans:**Is it possible in Mongo to search for documents in a collection that are missing a certain field?  
Yeah, it’s possible using $exists:

db.things.find( { a : { $exists : false } } ); // return if a is missing

If you don’t care if the field is missing or null (or if it’s never null) then you can use the slightly shorter and safer:

db.things.find( { a : null } ); // return if a is missing or null

It’s safer because $exists will return true even if the field is null, which often is not the desired result and can lead to an NPE.

**Q77) Is there a sample MongoDB Database along the lines of world for MySql?**

**Ans:**As someone new to Mongo, I am looking for a sample MongoDB database that I can import and play with. Something along the lines of world for mysql or Northwind for MSSQL.  
Is there one? (I couldn’t find any reference to one at https://www.mongodb.org nor did my googling help)  
For Windows users: Please follow the following steps to import from the json file if you are using windows 7:

1. Download the above mentionedJSON file and place it inside a folder (say d:sample)
2. Open a command prompt, start the mongo server by going in to the bin directory and typing in mongoD
3. Now take another command prompt and go to the bin directory again and write following command C:mongodbbin>mongoimport –db test –collection zips –file d:samplezips.json
4. The import would start working immediatly and at the end it would show some thing like this Thu Dec 19 17:11:22 imported 29470 objects

That’s it!  
I found this you can import the json data with:  
mongoimport –db scratch –collection zips –file zips.json  
I guess that you can import any json data that you find, it also supports csv and tsv. Hope this helps.  
This doesn’t have everything, but it is a nice step towards getting Northwind on MongoDB:  
https://github.com/tmcnab/northwind-mongo

**Q78) MongoDB query help: $elemMatch in nested objects**

**Ans:**

|  |  |
| --- | --- |
| 1  2  3  4  5  6 | > db.test.insert({“a” : { “b” : { “c” : { “d1” : [ “e1” ],  “d2” : [ “e2” ],  “d3” : [ “e3”, “e4” ],  “d4” : [ “e5”, “e6” ] } } } })  > db.test.find({‘a.b.c’ : {$exists : true}})  { “\_id” : ObjectId(“4daf2ccd697ebaacb10976ec”), “a” : { “b” : { “c” : { “d1” : [ “e1” ], “d2” : [ “e2” ], “d3” : [ “e3”, “e4” ], “d4” : [ “e5”, “e6” ] } } } } |

But none of these work:

|  |  |
| --- | --- |
| 1  2  3 | > db.test.find({‘a.b’: “c”})  > db.test.find({‘a.b’: {$elemMatch : {“c” : {$exists: true}}}})  > db.test.find({‘a.b’: {$elemMatch : {$elemMatch : {$all : [“e1”] }}}}) |

Suppose I don’t know what the values of c and d1…d4 are. Is there a generic way to search the nested-objects’ structure for particular values?  
I thought that was what $elemMatch was for.  
Thank you.

I thought that was what $elemMatch was for…  
From the docs: Using the $elemMatch query operator, you can match an entire document within an array.  
This does not sounds like what you’re looking for.

Is there a generic way to search the nested-objects’ structure for particular values?  
It sounds like you want to search “everything in object ‘c’ for an instance of ‘e1'”.  
MongoDB supports two related features, but the features not quite what you’re looking for.  
1. Reach into objects, dot notation:db.test.find({‘a.b.c.d1’ : ‘e1’})  
2. Read through arrays: `db.test.find({‘a.b.c.d4’ : ‘e5’})  
It sounds like you’re looking for the ability to do both at the same time. You want to “reach into objects” and “read through arrays” in the same query.  
Unfortunately, I do not know of such a feature. You may want to file a feature request for this.

**Related Page:**[**MongoDB Create Index**](https://mindmajix.com/mongodb-create-index)

**Q79) MongoDB Cursor Timeouts while doing a lot of writes**

**Ans:**We have a cluster of 2 replica Sets, with 3 servers per set. With a single collection being sharded. We also have a quite a few more(8+) collections that we use on a daily basis. With the majority of the data being in the sharded collection with close to 100 Million records in it.

Recently we have added the requirement to obtain 100x the data that we had been getting previously, and we need to write this to mongodb. A daemon has been set in place to perform the writes necessary to keep the database up to date. The script performs at over 200 writes a second, with the majority going to the all separate collections.  
With this quantity of writes, we have been unable to perform large reads for analytical purposes. Receiving a combination of Cursor Timeouts client-side and server-side(“Cursor Not Found”).

We have attempted to do limit/skip schemes on the reads, but the problem persists. What is the best course of action to remedy this as we require both a large amount of writes, with few, but large reads?

Typically, in a case like this you want to start looking at the queries causing the time. Then you want to look at the hardware to see what’s being stressed.

1. Are these queries correctly indexed?
2. How big are the indexes? Do they fit in RAM?
3. Can you provide some details on where the bottlenecks are?
4. Are you locked on on IO?
5. Are your processors running at full speed?

Also, is there anything unusual in the logs?  
Basically we need to ensure that you have: 1. Correctly built the system to handle the query 2. Correctly provisioned the system to handle the data volumes

**Q80) Is moving documents between collections a good way to represent state changes in MongoDB?**

**Ans:**I have two collections, one (**A)** containing items to be processed (relatively small) and one (**B**) with those already processed (fairly large, with extra result fields).  
Items are read from **A**, get processed and save()’d to **B**, then remove()’d from **A**.

The rationale is that indices can be different across these, and that the “incoming” collection can be kept very small and fast this way.  
I’ve run into two issues with this:

1. if either remove() or save() time out or otherwise fail under load, I lose the item completely, or process it twice
2. if both fail, the side effects happen but there is no record of that I can sidestep the double-failure case with findAndModify locks (not needed otherwise, we have a process-level lock) but then we have stale lock issues and partial failures can still happen. There’s no way to atomically remove+save to different collections, as far as I can tell (maybe by design?)  
   Is there a Best Practice for this situation?
3. There’s no way to atomically remove+save to different collections, as far as I can tell (maybe by design?)  
   Yes this is by design. MongoDB explicitly does not provides joins or transactions. Remove + Save is a form of transaction.  
   Is there a Best Practice for this situation?  
   You really have two low-complexity options here, both involve findAndModify.  
   **Option #1: a single collection**  
   Based on your description, you are basically building a queue with some extra features. If you leverage a single collection then you use findAndModify to update the status of each item as it is processing.  
   Unfortunately, that means you will lose this: …that the “incoming” collection can be kept very small and fast this way.  
   **Option #2: two collections**  
   The other option is basically a two phase commit, leveraging findAndModify.  
   Take a look at the docs for this here.  
   Once an item is processed in A you set a field to flag it for deletion. You then copy that item over toB. Once copied to B you can then remove the item from A.

**Q81) For Mongodb is it better to reference an object or use a natural String key?**

**Ans:**I am building a corpus of indexed sentences in different languages. I have a collection of Languages which have both an ObjectId and the ISO code as a key. Is it better to use a reference to the Language collection or store a key like “en” or “fr”?  
I suppose it’s a compromise between:

1. ease of referencing the Language
2. object in that collection
3. speed in doing queries where the sentence has a certain language
4. the size of the data on disk

Any best practices that I should know of?  
In the end, it really comes down to personal choice and what will work best for your application.  
The only requirement that MongoDB imposes upon \_id is that it be unique. It can be an ObjectId (which is provided by default), a string, even an embedded document (As I recall it cannot be an Array though).

In this case, you can likely guarantee ISO Code is a unique value and it may be an ideal value. You have a ‘known’ primary key which is also useful in itself by being identifiable, so using that instead of a generated ID is probably a more sensible bet. It also means anywhere you ‘reference’ this information in another collection you can save the ISO Code instead of an Object ID; those browsing your raw data can immediately identify what information that reference points at.

As an aside:  
The two big benefit of ObjectId is that they can be generated uniquely across multiple machines, processes and threads without needing any kind of central sequence tracking by the MongoDB server. They also are stored as a special type in MongoDB that only uses 12 bytes (as opposed to the 24 byte representation of the string version of an ObjectID)

**Q82) Does the MongoDB stats() function return bits or bytes?**

**Ans:**When using MongoDB’s .stats() function to determine document size, are the values returned in bits or bytes?  
Bytes of course. Unless you pass in a scale as optional argument.  
Running the collStats command – db.collection.stats() – returns all sizes in bytes, e.g.

|  |  |
| --- | --- |
| 1  2  3  4  5 | > db.foo.stats()  {  “size” : 715578011834,  // total size (bytes)  “avgObjSize” : 2862,    // average size (bytes)  } |

However, if you want the results in another unit then you can also pass in a scale argument.  
For example, to get the results in KB:

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11 | > db.foo.stats(1024)  {  “size” : 698806652,  // total size (KB)  “avgObjSize” : 2,    // average size (KB)  }  Or for MB:  > db.foo.stats(1024 \* 1024)  {  “size” : 682428,    // total size (MB)  “avgObjSize” : 0,   // average size (MB)  } |

**Q83) Storing Friend Relationships in MongoDB?**

**Ans:**I was wondering what the best way of storing friend relationship data using MongoDB is?  
Coming from mysql I had a separate table with friend relationships that had two foreign keys, each pointing to a friend in the “friendship” however, with MongoDB its possible to have arrays of references or even embedded documents..so whats the best way to store these relationships  
Immediate first reaction was that I’d store an array of friend object ids in each user, however, that troubles me, because then, in order to remove a “friendship” I would have to do deletes on both friend’s documents whereas if I stored the relationship in a separate collection (a la SQL) i could remove or add a relationship by just modifying one collection.  
Thanks!

Keeping a list of friend\_ids in a user, is what I’ll recommend. Few reasons,  
1.You query a user, and you have list of all friends upfront available.  
2.The requests (pending, accepted) can be handled well, by seeing that a respective ids should be present in both the user’s friends list. And I can get list of actual and accepted friends by querying

my\_id, my\_friends = user.\_id, user.friends<br>db.users.find({‘\_id’:{‘$in’: my\_friends}, ‘friends’: my\_id})

Yes, while removing a friendship , you have to $pull from both the friend’s list of both users, but frequency of that would be much less. But you query less in getting the friend-list, which would be used frequently.

**Q84) MongoDB – file size is huge and growing**

**Ans:**I have an application that use mongo for storing short living data. All data older than 45 minutes is removed by script something like:  
oldSearches = [list of old searches]  
connection = Connection()  
db = connection.searchDB  
res = db.results.remove{‘search\_id’:{“$in”:oldSearches}})  
I’ve checked current status –

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9 | >db.results.stats()  {  “ns” : “searchDB.results”,  “count” : 2865,  “size” : 1003859656,  “storageSize” : 29315124464,  “nindexes” : 1,  “ok” : 1  } |

So, according to this 1gb of data occupies 29GB of storage. Data folder looks like this(You may see that many files are very old – last accessed in the middle of may):  
Skip code block

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25  26 | ls -l /var/lib/mongodb/  total 31506556  -rwxr-xr-x 1 mongodb nogroup          6 2011-06-05 18:28 mongod.lock  -rw——- 1 mongodb nogroup   67108864 2011-05-13 17:45 searchDB.0  -rw——- 1 mongodb nogroup  134217728 2011-05-13 14:45 searchDB.1  -rw——- 1 mongodb nogroup 2146435072 2011-05-20 20:45 searchDB.10  -rw——- 1 mongodb nogroup 2146435072 2011-05-28 00:00 searchDB.11  -rw——- 1 mongodb nogroup 2146435072 2011-05-27 13:45 searchDB.12  -rw——- 1 mongodb nogroup 2146435072 2011-05-29 16:45 searchDB.13  -rw——- 1 mongodb nogroup 2146435072 2011-06-07 13:50 searchDB.14  -rw——- 1 mongodb nogroup 2146435072 2011-06-06 01:45 searchDB.15  -rw——- 1 mongodb nogroup 2146435072 2011-06-07 13:50 searchDB.16  -rw——- 1 mongodb nogroup 2146435072 2011-06-07 13:50 searchDB.17  -rw——- 1 mongodb nogroup 2146435072 2011-06-06 09:07 searchDB.18  -rw——- 1 mongodb nogroup  268435456 2011-05-13 14:45 searchDB.2  -rw——- 1 mongodb nogroup  536870912 2011-05-11 00:45 searchDB.3  -rw——- 1 mongodb nogroup 1073741824 2011-05-29 23:37 searchDB.4  -rw——- 1 mongodb nogroup 2146435072 2011-05-13 17:45 searchDB.5  -rw——- 1 mongodb nogroup 2146435072 2011-05-18 17:45 searchDB.6  -rw——- 1 mongodb nogroup 2146435072 2011-05-16 01:45 searchDB.7  -rw——- 1 mongodb nogroup 2146435072 2011-05-17 13:45 searchDB.8  -rw——- 1 mongodb nogroup 2146435072 2011-05-23 16:45 searchDB.9  -rw——- 1 mongodb nogroup   16777216 2011-06-07 13:50 searchDB.ns  -rw——- 1 mongodb nogroup   67108864 2011-04-23 18:51 test.0  -rw——- 1 mongodb nogroup   16777216 2011-04-23 18:51 test.ns  According to “top” mongod uses 29G of virtual memory ( and 780Mb of RSS) |

Why do i have such abnormal values? Do i need to run something additional to .remove() function to clean up database from old values?

**Virtual memory size** and resident size will appear to be very large for the mongod process. This is benign: virtual memory space will be just larger than the size of the datafiles open and mapped; resident size will vary depending on the amount of memory not used by other processes on the machine.

<a href="https://www.mongodb.org/display/DOCS/Caching">https://www.mongodb.org/display/DOCS/Caching</a>

When you remove an object from MongoDB collection, the space it occupied is not automatically garbage collected and new records are only appended to the end of data files, making them grow bigger and bigger. This explains it all:

<a href="https://www.mongodb.org/display/DOCS/Excessive+Disk+Space">https://www.mongodb.org/display/DOCS/Excessive+Disk+Space</a>

For starters, simply use:

db.repairDatabase()

**Q85) Mongo does not have a max() function, how do I work around this?**

I have a MongoDB collection and need to find the max() value of a certain field across all docs. This value is the timestamp and I need to find the latest doc by finding the largest timestamp. Sorting it and getting the first one gets inefficient really fast. Shall I just maintain a ‘maxval’ separately and update it whenever a doc arrives with a larger value for that field? Any better suggestions? Thanks much.

For sure if it will be big collection and if you need always display max timestamp you may need create separate collection and store statistic data there instead of order big collection each time.

|  |  |
| --- | --- |
| 1  2  3  4  5  6 | statistic  {  \_id = 1,  id\_from\_time\_stamp\_collection = ‘xxx’,  max\_timestamp: value  } |

And whenever new doc come just update statistic collection with id = 1(with $gt condition in query, so if new timestamp will be greater than max\_timestamp then max\_timestamp will be updated, otherwise – no).  
Also probably you can store and update other statistic data within statistic collection.  
if you have an index on the timestsamp field, finding the highest value is efficientl something like  
db.things.find().sort({ts:-1}).limit(1)  
but if having an index is too much overhead storing the max in a separate collection might be good.

**Q86) Mongodb drop every database**

**Ans:**I would like to know if there’re a command to drop every databases from my mongodb ?  
I know if I want to drop only one datatable, I just need to type the name of the database like the code below but I dont want to have to specify it.  
mongo DB\_NAME –eval ‘db.dropDatabase();’  
Any help would be appreciated.  
Thanks.  
John  
you can create a javascript loop that do the job and then execute it in the mongoconsole.

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17 | var dbs = db.getMongo().getDBNames()  for(var i in dbs){  db = db.getMongo().getDB( dbs[i] );  print( “dropping db ” + db.getName() );  db.dropDatabase();  }  save it to dropall.js and then execute:  mongo dropall.js  Try this command:  mongo –quiet –eval ‘db.getMongo().getDBNames().forEach(function(i){db.getSiblingDB(i).dropDatabase()})’  You can do it easy through c# official driver:  var \_mongoServer = MongoServer.Create(“mongodb://localhost:27020”);  var names = \_mongoServer.GetDatabaseNames();  foreach (var name in names)  {  \_mongoServer.DropDatabase(name);  } |

**Q87) mongodb find by comparing field values**

**Ans:**Is it possible to express the following SQL query in mongodb:  
SELECT \* FROM table AS t WHERE t.field1 > t.filed2;

**edit:** To summarize:.  
using a third field storing “field1 – field2” is almost perfect, but requires a little extra maintenance.  
$where will load and eval in JavaScript and won’t use any indexes. No good for large data.  
map/reduce has the same problem and will go trough all records even if we need only one  
You can do this using $where:  
db.coll.find( { $where: “this.field1 > this.field2” } );  
**But:**  
Javascript executes more slowly than the native operators, but it is very flexible  
If performance is an issue better to go with way suggested by @yi\_H.  
You could store in your document field1 – field2 as field3, then search for { field3: { $gt: 0 } }  
It also possible to get matching documents with mapreduce.

**Q88) insert to specific index for mongo array**

**Ans:**Mongo supports arrays of documents inside documents. For example, something like

|  |  |
| --- | --- |
| 1  2  3 | {\_id: 10, “coll”: [1, 2, 3] }  Now, imagine I wanted to insert an arbitrary value at an arbitrary index  {\_id: 10, “coll”: [1, {name: ‘new val’}, 2, 3] } |

I know you can update values in place with $ and $set, but nothing for insertion. it kind of sucks to have to replace the entire array just for inserting at a specific index.  
Now you can achieve this with a new upcoming version 2.6. To do this you have to use $positionoperator. For your particular example:

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9 | db.students.update(  { \_id: 10},  { $push: {  coll: {  $each: [ {name: ‘new val’} ],  $position: 1  }  }}  ) |

**Q89) Where does MongoDB store its documents?**

**Ans:**I have inserted and fetched data using MongoDB, in PHP. Is there an actual copy of this data in a document somewhere?  
MongoDB stores it’s data in the data directory specified by –dbpath. It uses a database format so it’s not actual documents, but there are multiple documents in each file and you cannot easily extract the data from this format yourself.

To read and/or update a document you need to use a MongoDB client, in the same way that you send SQL queries to MySQL through a MySQL client. You probably want to do it programmatically by using one of the client libraries for your programming language, but there is also a command-line client if you need to do manual updates.

By default Mongo stores its data in the directory /data/db.  
You can specify a different directory using the –dbpath option.  
If youre running Mongo on Windows it will be C:datadb (the driver letter will be the same as the working directory). To avoid confusion, Id recommend that you always specify a data directory using–dbpath.

**Q90) What does it mean to fit “working set” into RAM for MongoDB?**

**Ans:**MongoDB is fast, but only when your working set or index can fit into RAM. So if my server has 16G of RAM, does that mean the sizes of all my collections need to be less than or equal to 16G? How does one say “ok this is my working set, the rest can be “archived?”

“Working set” is basically the amount of data AND indexes that will be active/in use by your system.  
So for example, suppose you have 1 year’s worth of data. For simplicity, each month relates to 1GB of data giving 12GB in total, and to cover each month’s worth of data you have 1GB worth of indexes again totalling 12GB for the year.

If you are always accessing the last 12 month’s worth of data, then your working set is: 12GB (data) + 12GB (indexes) = 24GB.  
However, if you actually only access the last 3 month’s worth of data, then your working set is: 3GB (data) + 3GB (indexes) = 6GB. In this scenario, if you had 8GB RAM and then you started regularly accessing the past 6 month’s worth of data, then your working set would start to exceed past your available RAM and have a performance impact.  
But generally, if you have enough RAM to cover the amount of data/indexes you expect to be frequently accessing then you will be fine.  
Edit: Response to question in comments

I’m not sure I quite follow, but I’ll have a go at answering. Firstly, the calculation for working set is a “ball park figure”. Secondly, if you have a (e.g.) 1GB index on user\_id, then only the portion of that index that is commonly accessed needs to be in RAM (e.g. suppose 50% of users are inactive, then 0.5GB of the index will be more frequently required/needed in RAM). In general, the more RAM you have, the better especially as working set is likely to grow over time due to increased usage. This is where sharding comes in – split the data over multiple nodes and you can cost effectively scale out. Your working set is then divided over multiple machines, meaning the more can be kept in RAM. Need more RAM? Add another machine to shard on to.

The working set is basically the stuff you are using most (frequently). If you use index A for collection B to search for a subset of documents then you could consider that your working set. As long as the most commonly used parts of those structures can fit in memory then things will be exceedingly fast. As parts no longer fit in your working set, like many of the documents then that can slow down. Generally things will become much slower if your indexes exceed your memory.

Yes, you can have lots of data, where most of it is “archived” and rarely used without affecting the performance of our application or impacting your working set (which doesn’t include that archived data).

**Q91) MongoDB: Updating a document in an array**

**Ans:**I have a collection with documents of this schema:

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12 | {  \_id: something,  recipients: [{id:1, name:”Andrey”, isread:false}, {id:2, name:”John”, isread:false}]  }  Now, I want to update “isread” for John (id = 2) using findAndModify(), because I also need to get the original document.  I’m trying this command:  db.messages.findAndModify({query:{‘recipients.id’:2}, update:{‘recipients.$.isread’:true}})  but what it does, it just replaces the whole “recipients” field with ‘recipients.$.isread’, so the document now looks like:  {  \_id: someid,  ‘recipients.$.isread’:true  } |

What am I doing wrong?  
Try to use $set like this:

db.messages.findAndModify({query:{‘recipients.id’:2}, update:{$set:{‘recipients.$.isread’:true}}})

**Q92) How to limit number of updating documents in mongodb**

**Ans:**How to implement somethings similar to db.collection.find().limit(10) but while updating documents?  
Now I’m using something really crappy like getting documents withdb.collection.find().limit() and then updating them.  
In general I wanna to return given number of records and change one field in each of them.  
Thanks.

Unfortunately the workaround you have is the only way to do it AFAIK. There is a boolean flagmulti which will either update all the matches (when true) or update the 1st match (when false).

You can use:

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7 | db.collection.find().limit(NUMBER\_OF\_ITEMS\_YOU\_WANT\_TO\_UPDATE).forEach(  function (e) {  e.fieldToChange = “blah”;  ….  db.collection.save(e);  }  ); |

(Credits for forEach code: MongoDB: Updating documents using data from the same document)  
What this will do is only change the number of entries you specify. So if you want to add a field called “newField” with value 1 to only half of your entries inside “collection”, for example, you can put in

|  |  |
| --- | --- |
| 1  2  3  4  5  6 | db.collection.find().limit(db.collection.count() / 2).forEach(  function (e) {  e.newField = 1;  db.collection.save(e);  }  ); |

If you then want to make the other half also have “newField” but with value 2, you can do an update with the condition that newField doesn’t exist:  
db.collection.update( { newField : { $exists : false } }, { $set : { newField : 2 } }, {multi : true} );

**Q93) Compound indices for OR+Sort query in mongodb**

**Ans:**For this OR query:

db.messages.find({ $or: [ { to: { $ne: null }, from: “xyz” }, { to: “xyz” } ] }).sort({\_id: -1}).limit(50)

with following indices:

{to:1, from: 1, \_id:-1} and {from:1, to:1, \_id:-1}

mongo is always doing a full scan.  
I was hoping that mongo could use these two indices and merge the results.  
Do I need to split this into two queries (one for each OR clause) and merge myself? Or is there some other index that can help?

This is a known issue, https://jira.mongodb.org/browse/SERVER-1205, which you can vote for if it is very important for your use case.

**Q94) MongoDB: Unique index on array element’s property**

**Ans:**I have a structure similar to this:

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7 | class Cat {  int id;  List kittens;  }  class Kitten {  int id;  } |

I’d like to prevent users from creating a cat with more than one kitten with the same id. I’ve tried creating an index as follows:  
db.Cats.ensureIndex({‘id’: 1, ‘kittens.id’: 1}, {unique:true})  
But when I attempt to insert a badly-formatted cat, Mongo accepts it.  
Am I missing something? can this even be done?  
As far as I know, unique indexes only enforce uniqueness across different documents, so this would throw a duplicate key error:

|  |  |
| --- | --- |
| 1  2 | db.cats.insert( { id: 123, kittens: [ { id: 456 } ] } )  db.cats.insert( { id: 123, kittens: [ { id: 456 } ] } ) |

But this is allowed:

db.cats.insert( { id: 123, kittens: [ { id: 456 }, { id: 456 } ] } )

I’m not sure if there’s any way enforce the constraint you need at the Mongo level, maybe it’s something you could check in the application logic when you insert of update?

**Q95) Updating embedded document property in Mongodb**

**Ans:**I have a document that looks like this:  
Skip code block

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11 | {  “\_id”: 3,  “Slug”: “slug”,  “Title”: “title”,  “Authors”: [  {  “Slug”: “slug”,  “Name”: “name”  }  ]  } |

I want to update all Authors.Name based on Authors.Slug. I tried this but it didn’t work:

.update({“Authors.Slug”:”slug”}, {$set: {“Authors.Name”:”zzz”}});

What am I doing wrong here?

.update(Authors:{$elemMatch:{Slug:”slug”}}, {$set: {‘Authors.$.Name’:”zzz”}});

**Q96) What is the best way to store dates in MongoDB?**

**Ans:**I am just starting to learn about MongoDB and hoping to slowly migrate from MySQL.

In MySQL, there are two different data types – DATE (‘0000-00-00’) and DATETIME (‘0000-00-00 00:00:00’). In my MySQL, I use the DATE type, but I am not sure how to transfer them into MongoDB. In MongoDB, there is a Date object, which is comparable to DATETIME. It seems it would be most appropriate to use Date objects, but that would be wasting space, since hours, min, sec are not utilized. On the other hand, storing dates as strings seems wrong.

Is there a golden standard on storing dates (‘0000-00-00’) in MongoDB?

BSON (the storage data format used by mongo natively) has a dedicated date type UTC datetime which is a 64 bit (so, 8 byte) signed integer denoting milliseconds since Unix time epoch. There are very few valid reasons why you would use any other type for storing dates and timestamps.

If you’re desperate to save a few bytes per date (again, with mongo’s padding and minimum block size and everything this is only worth the trouble in very rare cases) you can store dates as a 3 byte binary blob by storing it as an unsigned integer in YYYYMMDD format, or a 2 byte binary blob denoting “days since January 1st of year X” where X must be chosen appropriately since that only supports a date range spanning 179 years.

EDIT: As the discussion below demonstrates this is only a viable approach in very rare circumstances. **Basically; use mongo’s native date type**  
I’m actually in the process of converting a MongoDB database where dates are stored as proper Date() types to instead store them as strings in the form yyyy-mm-dd. Why, considering that every other answerer says that this is a horrible idea? Simply put, because of the neverending pain I’ve been suffering trying to work with dates in JavaScript, which has no (real) concept of timezones.

I had been storing UTC dates in MongoDB, i.e. a Date() object with my desired date and the time set as midnight UTC, but it’s unexpectedly complicated and error-prone to get a user submitted date correctly converted to that from whatever timezone they happen to be in. I’ve been struggling to get my JavaScript “whatever local timezone to UTC” code to work (and yes, I’m aware of Sugar.js and Moment.js) and I’ve decided that simple strings like the good old MySQL standard yyyy-mm-dd is the way to go, and I’ll parse into Date() objects as needed at runtime on the client side.

Incidentally, I’m also trying to sync this MongoDB database with a FileMaker database, which also has no concept of timezones. For me the simplicity of simply not storing time data, especially when it’s meaningless like UTC midnight, helps ensure less-buggy code even if I have to parse to and from the string dates now and then.

**Q97) MongoDb : Avoid excessive disk space**

**Ans:**I am having a database which has been allocated a space of 85GB. I got this size using the show dbs command. But when I use the db.stats(), i get the storage size as 63GB. After going through the docs I found dat mongo db allocates a size to the db that is created and then the actual data is filled.

Why does mongo does this preallocation and is there a way I can avoid this preallocation ?  
If Yes, will it be a good idea to do it or it is going to affect from performance point of view ?  
Thanks!!!

Details can be found here: https://www.mongodb.org/display/DOCS/Excessive+Disk+Space  
Now to your case. I think that the current size of your datafiles cannot be caused by preallocation alone. The maximum size of allocated data files is 2G. Maybe you once stored much more data than today? The link above also shows how to reclaim this data if you’re sure you will not need it soon anyway. This will require a lot of disk space during the process, so you must give this some planning.

Disabling preallocation will indeed have perfomance impact. When you do many inserts, the process would regularly have to wait to create a new file.

**Q98) Get n-th element of an array in mongo**

**Ans:**As part of my document in mongo I’m storing an array of objects. How can I query it for only the 4th element of the array for example? So I don’t want the get the entire array out, just the 4th element.  
Use $slice.

db.foo.find({ bar : “xyz” } , { my\_array : { $slice : [n , 1] } } )

will retrieve the nth element of the array “my\_array” of all documents in the foo collection where bar = “xyz”.  
Some other examples from the MongoDB documentation:

|  |  |
| --- | --- |
| 1  2  3  4 | db.posts.find({}, {comments:{$slice: 5}}) // first 5 comments  db.posts.find({}, {comments:{$slice: -5}}) // last 5 comments  db.posts.find({}, {comments:{$slice: [20, 10]}}) // skip 20, limit 10  db.posts.find({}, {comments:{$slice: [-20, 10]}}) // 20 from end, limit 10 |

Which you can read here: <a href="https://www.mongodb.org/display/DOCS/Retrieving+a+Subset+of+Fields">https://www.mongodb.org/display/DOCS/Retrieving+a+Subset+of+Fields</a>

Another way to do this is to use the update array syntax. Here, contribs.1 sets the second element in the contribs array to have value ALGOL 58 (Taken from the manual page on update syntax)

|  |  |
| --- | --- |
| 1  2  3  4 | db.bios.update(  { \_id: 1 },  { $set: { ‘contribs.1’: ‘ALGOL 58’ } }  ) |

**Q99) How to delete n-th element of array in mongodb**

**Ans:**For example i have a document  
db.test.save({\_id: 1, list:[  
{key: “a”},  
{key: “b”},  
{key: “c”},  
{key: “d”},  
{key: “e”}  
]})  
and i need remove second element from the list. For now I do that in a two steps. First of all I unset second list element but unset operator do not remove element, it is going to be null, after that I pull any nullable value from the list field

|  |  |
| --- | --- |
| 1  2 | db.test.update({\_id: 1}, {$unset: {“list.2”: 1}})  db.test.update({\_id: 1}, {$pull: {list: null}}) |

I want to ask whether there is solution do that in a one operation?  
No, unfortunately what you are doing is currently the best option. Have a look at this question: In mongoDb, how do you remove an array element by it’s index which links to a Jira for this very issue.

**Q100) Slow pagination over tons of records in mongo**

**Ans:**I have over 300k records in one collection in Mongo.  
When I run this very simple query:  
db.myCollection.find().limit(5);  
It took only few miliseconds. When I use skip in the query:  
db.myCollection.find().skip(200000).limit(5)  
It won’t return anything… it is running minutes and return nothing.  
How to make it better?  
From MongoDB documentation:

**Paging Costs**

Unfortunately skip can be (very) costly and requires the server to walk from the beginning of the collection, or index, to get to the offset/skip position before it can start returning the page of data (limit). As the page number increases skip will become slower and more cpu intensive, and possibly IO bound, with larger collections.  
Range based paging provides better use of indexes but does not allow you to easily jump to a specific page.

You have to ask yourself a question: how often do you need 40000th page? Also see this article;  
One approach to this problem, if you have large quantities of documents and you are displaying them in sorted order (I’m not sure how useful skip is if you’re not) would be to use the key you’re sorting on to select the next page of results.

So if you start with

db.myCollection.find().limit(100).sort(created\_date:true);

and then extract the created date of the last document returned by the cursor into a variablemax\_created\_date\_from\_last\_result, you can get the next page with the far more efficient (presuming you have an index on created\_date) query

db.myCollection.find({created\_date : { $gt : max\_created\_date\_from\_last\_result} }).limit(100).sort(created\_date:true);

**Q101) How to use variables in MongoDB Map-reduce map function**

**Ans:**Given a document

{\_id:110000, groupings:{A:’AV’,B:’BV’,C:’CV’,D:’DV’},coin:{old:10,new:12}}

My specs call for the specification of attributes for mapping and aggregation at run time, as the groupings the interested in are not known up front, but specified by the user.  
For example, one user would specify [A,B] which will cause mapping emissions of

emit( {A:this.groupings.A,B:this.groupings.B},this.coin )

while another would want to specify [A,C] which will cause mapping emissions of

emit( {A:this.groupings.A,C:this.groupings.C},this.coin )

**[MongoDB Certification Training!](https://mindmajix.com/mongodb-training" \l "curriculum" \t "_blank)**

[Explore Curriculum](https://mindmajix.com/mongodb-training" \l "curriculum" \t "_blank)

B/c the mapper and reducer functions execute server side, and don’t have access to client variables, I haven’t been able to come up with a way to use a variable map key in the mapper function.

If I could reference a list of things to group by from the scope of the execution of the map function, this is all very strait forward. However, b/c the mapping function end up getting from from a different scope, I don’t know how to do this, or if it’s even possible.

Before I start trying to dynamically build java script to execute through the driver, does anyone have a better suggestion? Maybe a ‘group’ function will handle this scenario better?  
You can pass global, read-only data into map-reduce functions using the “scope” parameter on the map-reduce command. It’s not very well documented, I’m afraid. As pointed out by @Dave Griffith, you can use the scope parameter of the mapReduce function.

I struggled a bit to figure out how to properly pass it to the function because, as pointed out by others, the documentation is not very detailed. Finally, I realised that mapReduce is expecting 3 params:  
1. map function  
2. reduce function  
3. object with one or more of the params definedin the doc

Eventually, I arrived at the following code in Javascript:

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25  26  27  28  29  30 | // I define a variable external to my map and to my reduce functions  var KEYS = {STATS: “stats”};  function m() {  // I use my global variable inside the map function  emit(KEYS.STATS, 1);  }  function r(key, values) {  // I use a helper function  return sumValues(values);  }  // Helper function in the global scope  function sumValues(values) {  var result = 0;  values.forEach(function(value) {  result += value;  });  return value;  }  db.something.mapReduce(  m,  r,  {  out: {inline: 1},  // I use the scope param to pass in my variables and functions  scope: {  KEYS: KEYS,  sumValues: sumValues // of course, you can pass function objects too  }  }  ); |

**Q102) mongo dbname –eval ‘db.collection.find()’ does not work**

**Ans:**Why does this work:

|  |  |
| --- | --- |
| 1  2  3  4  5  6 | # mongo dbname  MongoDB shell version: 1.8.3  connecting to: nextmuni\_staging  > db.collection.find()  { “foo” : “bar” }  > bye |

While this does not work:

|  |  |
| --- | --- |
| 1  2  3  4 | # mongo localhost/dbname –eval ‘db.collection.find()’  MongoDB shell version: 1.8.3  connecting to: localhost/dbname  DBQuery: dbname.collection -> undefined |

It should be exactly the same, no?  
Thanks!  
The return val of db.collection.find() is a cursor type. Executing this command from within the shell will create a cursor and show you the first page of data. You can start going through the rest by repeating the ‘it’ command.

I think the scope of variables used during the execution of an eval’d script is only for the lifetime of the script (data can be persisted into collections of course) so once the script terminates those cursor variables no longer exist and so you would be able to send another eval script to page the data. So the behaviour you get during a shell session wouldn’t really work from an eval script.

To get close to the behaviour you could run something like this:  
mongo dbname –eval “db.collection.find().forEach(function(x){printjson(x);})”  
That shows you that the command does execute and produce a cursor which you can then iterate over sending the output to stdout.  
Edit: I think the point I was trying to make was that the command you are issuing is working its just the output is not what you expect.

**Q103) Compact command not freeing up space in MongoDB 2.0**

**Ans:**I just installed MongoDB 2.0 and tried to run the compact command instead of the repair command in earlier versions. My database is empty at the moment, meaning there is only one collection with 0 entries and the two system collections (indices, users). Currently the db takes about 4 GB of space on the harddisk. The db is used as a temp queue with all items being removes after they have been processed.

I tried to run the following in the mongo shell.  
use mydb  
db.theOnlyCollection.runCommand(“compact”)  
It returns with  
ok: 1  
But still the same space is taken on the harddisk. I tried to compact the system collections as well, but this did not work.  
When I run the normal repair command  
db.repairDatabase()  
the database is compacted and only takes 400 MB.  
Anyone has an idea why the compact command is not working?  
Thanks a lot for your help.  
Best  
Alex  
Collection compaction is not supposed to decrease the size of data files. Main point is to defragment collection and index data – combine unused space gaps into continuous space allowing new data to be stored there. Moreover it may actually increase the size of data files:  
Compaction may increase the total size of your data files by up to 2GB. Even in this case, total collection storage space will decrease.

https://www.mongodb.org/display/DOCS/compact+Command#compactCommand-Effectsofacompaction

**Q104) MongoDB: order by two fields sum**

**Ans:**SELECT (a+b) as c FROM my\_table ORDER BY c ASC;  
How can I order by two columns sum in Mongo?  
You can’t do it easy without an extra action.  
To sort on any computed value you need to store it in a document first or in other worlds you need to create extra field ‘c’, and store a + b in it with each update/insert and only then sort on ‘c’ as usual.

**Q105) Is it practical to use Mongo’s capped collection a memory cache?**

**Ans:**Is it practical to use Mongo’s capped collections as poor man’s memcache assuming it will be kept in memory most of the time?  
You can definitely use capped collections for this purpose. But it’s important to know the basic limitations.

1. Data will expire based on insert order not time-to-expire  
2. Data that is frequently accessed may still be pushed out of memory  
3. \_idcolumns are not defined on Capped Collections by default, you will need to ensure those indexes.  
4. The objects cannot be modified in a way that changes the size of the object. For example, youcouldincrement an existing integer, but you cannot add a field or modify a string value on the entry.  
5. The capped collection scannot be sharded.

Because of #1 & #4 & #5, you are definitely losing some the core Memcache functionality.  
There is a long-outstanding JIRA ticket for TTL-based capped collections which is probably exactly what you want.  
Of course, the big question in this whole debate is “where is the extra RAM”. Many people who use MongoDB as their primary store simply drop Memcache. If you have a bunch of extra RAM sitting around, why not just use it store the actual data instead of copies of that data?

**Q106) MongoDB Path Change?**

**Ans:**My server went down from an electrical failure and for a few horrifying seconds, I thought I’d lost all MongoDB data. I then realized that when the server restarted, mongo automatically restarted without the –dbpath option.

What I can’t figure out is why, even though my mongodb.conf has the dbpath set to /var/lib/mongodb, mongo looked for the db files in /data/db on startup. Can anyone shed some light on this?  
Thanks!

/data/db is the default path that mongod will look for data files in if it is started with no options. Does your startup script instruct mongod to load the correct config file? If not, that would explain this behavior.

When was the last time you updated Mongod on your system? and how did you update it?

Depending how you installed / updated Mongod it could happen that either the /etc/init.d/mongo script or the /etc/mongo.conf files could be overwritten.  
Or somebody edited those files a long time ago, and this is the first time Mongo got restarted, and now it looks like there’s suddenly a change.  
I haven’t seen either of those two things happen in a long time (but if you’re using Gentoo, you would probably see this happen)  
What OS do you use?  
It’s a good idea to keep backups of those files…

**Q107) In MongoDB, how do I find documents where array size is greater than 1?**

**Ans:**I have a MongoDB collection (accommodations) that has documents in the following format:

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16 | Skip code block  {  “\_id” : ObjectId(“4e8ae86d08101908e1000001”),  “name” : [“Some Name”],  “zipcode” : [“2223”]  }  {  “\_id” : ObjectId(“4e8ae86d08101908e1000002”),  “name” : [“Another “, “Name”],  “zipcode” : [“2224”]  }  {  “\_id” : ObjectId(“4e8ae86d08101908e1000003”),  “name” : [“Yet”, ” Another “, “Name”],  “zipcode” : [“2225”]  } |

My question is: How can I select all documents that have a name array size of greater than 1?  
I can get documents that match an array size:

db.accommodations.find({“name”:{$size:2 }})

This will correctly return the documents where there are 2 elements in the name array. However I can’t do a $gt command to return all documents where the name field has an array size of greater than 2:

db.accommodations.find({“name”:{$size: {$gt:1} }})

Any ideas how I can do this on existing data… preferably without modifying the data?  
Many thanks

**Update:**

For mongodb versions 2.2+ more efficient way to do this described by @JohnnyHK in anotheranswer.  
1.Using $where  
db.accommodations.find( { $where: “this.name.length > 1” } );  
But…  
Javascript executes more slowly than the native operators listed on this page, but is very flexible. See the server-side processing page for more information.

2.Create **extra**field NamesArrayLength, update it with names array length and then use in queries:  
db.accommodations.find({“NamesArrayLength”: {$gt: 1} });  
It will be better solution, and will work much faster (you can create index on it).  
There’s a more efficient way to do this in MongoDB 2.2+ now that you can use numeric array indexes in query object keys.  
// Find all docs that have at least a second name array element.  
db.accommodations.find({‘name.1’: {$exists: true}})  
I believe this is the fastest query that answers your question, because it doesn’t use an interpreted$where clause:

|  |  |
| --- | --- |
| 1  2  3  4  5 | {$nor: [  {name: {$exists: false}},  {name: {$size: 0}},  {name: {$size: 1}}  ]} |

It means “all documents except those without a name (either non existant or empty array) or with just one name.”  
Test:  
Skip code block

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11 | > db.test.save({})  > db.test.save({name: []})  > db.test.save({name: [‘George’]})  > db.test.save({name: [‘George’, ‘Raymond’]})  > db.test.save({name: [‘George’, ‘Raymond’, ‘Richard’]})  > db.test.save({name: [‘George’, ‘Raymond’, ‘Richard’, ‘Martin’]})  > db.test.find({$nor: [{name: {$exists: false}}, {name: {$size: 0}}, {name: {$size: 1}}]})  { “\_id” : ObjectId(“511907e3fb13145a3d2e225b”), “name” : [ “George”, “Raymond” ] }  { “\_id” : ObjectId(“511907e3fb13145a3d2e225c”), “name” : [ “George”, “Raymond”, “Richard” ] }  { “\_id” : ObjectId(“511907e3fb13145a3d2e225d”), “name” : [ “George”, “Raymond”, “Richard”, “Martin” ] }  > |

**Q108) heterogeneous bulk update in mongodb**

**Ans:**I know that we can bulk update documents in mongodb with  
db.collection.update( criteria, objNew, upsert, multi )  
in one db call, but it’s homogeneous, i.e. all those documents impacted are following one kind of criteria. But what I’d like to do is something like  
db.collection.update([{criteria1, objNew1}, {criteria2, objNew2}, …]  
, to send multiple update request which would update maybe absolutely different documents or class of documents in single db call.  
What I want to do in my app is to insert/update a bunch of objects with compound primary key, if the key is already existing, update it; insert it otherwise.  
Can I do all these in one combine in mongodb?  
That’s two seperate questions. To the first one; there is no MongoDB native mechanism to bulk send criteria/update pairs although technically doing that in a loop yourself is bound to be about as efficient as any native bulk support.  
Checking for the existence of a document based on an embedded document (what you refer to as compound key, but in the interest of correct terminology to avoid confusion it’s better to use the mongo name in this case) and insert/update depending on that existence check can be done with upsert :

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8 | document A :  {  \_id: ObjectId(…),  key: {  name: “Will”,  age: 20  }  } |

db.users.update({name:”Will”, age:20}, {$set:{age: 21}}), true, false)  
This upsert (update with insert if no document matches the criteria) will do one of two things depending on the existence of document A :  
1. Exists : Performs update “$set:{age:21}” on the existing document  
2. Doesn’t exist : Create a new document with fields “name” and field “age” with values “Will” and “20” respectively (basically the criteria are copied into the new doc) and then the update is applied ($set:{age:21}). End result is a document with “name”=”Will” and “age”=21.  
Hope that helps

**Q109) Is it possible to mongodump the last “x” records from a collection?**

**Ans:**Can you use mongodump to dump the latest “x” documents from a collection? For example, in the mongo shell you can execute:  
db.stats.find().sort({$natural:-1}).limit(10);  
Is this same capability available to mongodump?  
I guess the workaround would be to dump the above documents into a new temporary collection and mongodump the entire temp collection, but would be great to just be able to do this via mongodump.  
Thanks in advance,

Michael  
mongodump does not fully expose the cursor interfaces. But you can work around it, using the –query parameter. First get the total number of documents of the collection  
db.collection.count()  
Let’s say there are 10000 documents and you want the last 1000. To do so get the id of first document you want to dump.

db.collection.find().sort({\_id:1}).skip(10000 – 1000).limit(1)

In this example the id was “50ad7bce1a3e927d690385ec”. Now you can feed mongodump with this information, to dump all documents a with higher or equal id.  
$ mongodump -d ‘your\_database’ -c ‘your\_collection’ -q ‘{\_id: {$lte: ObjectId(“50ad7bce1a3e927d690385ec”)}}’

**UPDATE** the new parameters ‘–limit’ and ‘–skip’ were added to mongoexport will be probably available in the next version of the tool: https://github.com/mongodb/mongo/pull/307  
mongodump supports the –query operator. If you can specify your query as a json query, you should be able to do just that.

If not, then your trick of running a query to dump the records into a temporary collection and then dumping that will work just fine. In this case, you could automate the dump using a shell script that calls a mongo with a javascript command to do what you want and then calling mongodump.

**Q110) How do you get the size of an individual index in MongoDB?**

**Ans:**I know I can use db.collection.totalIndexSize() to get the total index size, but I’m interested in seeing the size of an individual index.  
Is this supported?

Certainly can. db.collection.stats().indexSizes is an embedded document where each index name is a key and the value is the total index size in bytes :

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10 | > db.test.stats()  {  “ns” : “test.test”,    “indexSizes” : {  “\_id\_” : 137904592,  “a\_1” : 106925728  },  “ok” : 1  } |

**Q111) mongodb Mongod complains that there is no /data/db folder**

**Ans:**I am using my new mac for the first time today. I am following the get started guide on the mongodb.org up until the step where one creates the /data/db directory. btw, I used the homebrew route.

So I open a terminal, and I think I am at what you called the Home Directory, for when I do “ls”, I see folders of Desktop Application Movies Music Pictures Documents and Library.  
So I did a  
mkdir -p /data/db  
first, it says permission denied. I kept trying different things for half and hour and finally :  
mkdir -p data/db  
worked. and when I “ls”, a directory of data and nested in it a db folder do exist.  
then I fire up mongod and it complains about not finding data/db  
Have I done something wrong?  
Now I have done the  
sudo mkdir -p /data/db  
and when I do a “ls” I do see the data dir and the db dir. inside the db dir though, there is absolutely nothing in it and when I now run mongod

Skip code block

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16 | Sun Oct 30 19:35:19 [initandlisten] exception in initAndListen: 10309 Unable to create/openlock file: /data/db/mongod.lock errno:13 Permission denied Is a mongod instance already running?, terminating  Sun Oct 30 19:35:19 dbexit:  Sun Oct 30 19:35:19 [initandlisten] shutdown: going to close listening sockets…  Sun Oct 30 19:35:19 [initandlisten] shutdown: going to flush diaglog…  Sun Oct 30 19:35:19 [initandlisten] shutdown: going to close sockets…  Sun Oct 30 19:35:19 [initandlisten] shutdown: waiting for fs preallocator…  Sun Oct 30 19:35:19 [initandlisten] shutdown: lock for final commit…  Sun Oct 30 19:35:19 [initandlisten] shutdown: final commit…  Sun Oct 30 19:35:19 [initandlisten] shutdown: closing all files…  Sun Oct 30 19:35:19 [initandlisten] closeAllFiles() finished  Sun Oct 30 19:35:19 [initandlisten] shutdown: removing fs lock…  Sun Oct 30 19:35:19 [initandlisten] couldn’t remove fs lock errno:9 Bad file descriptor  Sun Oct 30 19:35:19 dbexit: really exiting now  EDIT Getting error message for  sudo chown mongod:mongod /data/db  chown: mongod: Invalid argument |

Thanks, everyone!  
**You created the directory in the wrong place**

/data/db means that it’s directly under the ‘/’ root directory, whereas you created ‘data/db’ (without the leading /) probably just inside another directory, such as the ‘/root’ homedirectory.

**You need to create this directory as root**

Either you need to use sudo , e.g. sudo mkdir -p /data/db  
Or you need to do su – to become superuser, and then create the directory with mkdir -p /data/db

Note:  
MongoDB also has an option where you can create the data directory in another location, but that’s generally not a good idea, because it just slightly complicates things such as DB recovery, because you always have to specify the db-path manually. I wouldn’t recommend doing that.

Edit:  
the error message you’re getting is “U**nable to create/open lock file: /data/db/mongod.lock errno:13 Permission denied**”. The directory you created doesn’t seem to have the correct permissions and ownership — it needs to be writable by the user who runs the MongoDB process.  
To see the permissions and ownership of the ‘/data/db/’ directory, do this: (this is what the permissions and ownership should look like)  
$ ls -ld /data/db/  
drwxr-xr-x 4 mongod mongod 4096 Oct 26 10:31 /data/db/  
The left side ‘drwxr-xr-x’ shows the permissions for the User, Group, and Others. ‘mongod mongod’ shows who owns the directory, and which group that directory belongs to. Both are called ‘mongod’ in this case.

**If your ‘/data/db’ directory doesn’t have the permissions and ownership above, do this:  
sudo chmod 0755 /data/db**  
sudo chown mongod:mongod /data/db  
that should make it work..

Check here to better understand the meaning of the directory permissions:

<a href="https://www.dartmouth.edu/~rc/help/faq/permissions.html">https://www.dartmouth.edu/~rc/help/faq/permissions.html</a>

Maybe also check out one of the tutorials you can find via Google: “UNIX for beginners”  
After getting the same error as Nik  
chown: id -u: Invalid argument  
I found out this apparently occurred from using the wrong type of quotation marks (should have been **backquotes**) Ubuntu Forums  
Instead I just used  
sudo chown $USER /data/db  
as an alternative and now mongod has the permissions it needs.

**Q112) mongodb replicaset host name change error**

**Ans:**I have a mongodb replicaset on ubuntu.. In replica set, hosts are defined as localhost. You can see ;

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20 | Skip code block  {  “\_id” : “myrep”,  “version” : 4,  “members” : [  {  “\_id” : 0,  “host” : “localhost:27017”  },  {  “\_id” : 2,  “host” : “localhost:27018”  },  {  “\_id” : 1,  “host” : “localhost:27019”,  “priority” : 0  }  ]  } |

I want to change host adresses with real ip of server. But when i run rs.reconfig, I get error :

|  |  |
| --- | --- |
| 1  2  3  4  5  6 | {  “assertion” : “hosts cannot switch between localhost and hostname”,  “assertionCode” : 13645,  “errmsg” : “db assertion failure”,  “ok” : 0  } |

How can i solve it ? Thank you.  
There is a cleaner way to do this:  
use local

|  |  |
| --- | --- |
| 1  2  3 | cfg = db.system.replset.findOne({\_id:”replicaSetName”})  cfg.members[0].host=”newHost:27017?  db.system.replset.update({\_id:”replicaSetName”},cfg) |

then restart mongo

**Q113) Mongo compound indexes, using less-than-all in a query**

**Ans:**I understand that with MongoDB, for a query to make use of a compound index it must use ALL of the keys in the index, or at least some of the keys starting from the left. For example  
db.products.find({ “a”:”foo”, “b”:”bar” })  
Will happily make use of an index made up of {a, b, c}.  
However, if I want to query:  
db.products.find( {“a”:”foo”, “c”:”thing” })  
I believe this can’t use the index. Could this be solved by adding a trivial condition on “b”, e.g.  
db.products.find( {“a”:”foo”, “b”:{ $ne : “” }, “c”:”thing” })  
Even when I don’t actually care about the value of b. The reason for this is that we currently have 45m objects, and it’s going to continue growing so we’re looking to consolidate our indexes to save on resources.  
Many thanks.

In general, a query on a multi-column index that does not sufficiently limit matches for one of the columns will restrict the usefulness of the multi-column index. In your example, using query criteria of {“a”:”foo”, “b”:{$ne:””}, “c”:”thing”} will limit the usefulness of an {a,b,c} index to matching only on a. If your query criteria will be executed often, create an {a,c,b} index (or {a,c} ifb will not be used in query criteria).

Use the explain function on your queries to see if an index is being used to its full potential. Ifexplain tells you indexOnly is true, then your query is only using the index to find matching documents; otherwise, MongoDB needs to look at each document to find matches.  
For further information, see:  
1. Optimization  
2. Indexes  
3. Query Optimizer

**Q114) New to MongoDB Can not run command mongo**

**Ans:**I was trying to run MongoDB:

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25  26  27  28  29  30  31  32 | Skip code block  E:mongobin>mongod  mongod –help for help and startup options  Sun Nov 06 18:48:37  Sun Nov 06 18:48:37 warning: 32-bit servers don’t have journaling enabled by default. Please use –journal if you want durability.  Sun Nov 06 18:48:37  Sun Nov 06 18:48:37 [initandlisten] MongoDB starting : pid=7108 port=27017 dbpath=/data/db 32-bit host=pykhmer-PC  Sun Nov 06 18:48:37 [initandlisten]  Sun Nov 06 18:48:37 [initandlisten] \*\* NOTE: when using MongoDB 32 bit, you are limited to about 2 gigabytes of data  Sun Nov 06 18:48:37 [initandlisten] \*\*       see https://blog.mongodb.org/post/137788967/32-bit-limitations  Sun Nov 06 18:48:37 [initandlisten] \*\*       with –journal, the limit is lower  Sun Nov 06 18:48:37 [initandlisten]  Sun Nov 06 18:48:37 [initandlisten] db version v2.0.1, pdfile version 4.5  Sun Nov 06 18:48:37 [initandlisten] git version: 3a5cf0e2134a830d38d2d1aae7e88cac31bdd684  Sun Nov 06 18:48:37 [initandlisten] build info: windows (5, 1, 2600, 2, ‘Service Pack 3’) BOOST\_LIB\_VERSION=1\_42  Sun Nov 06 18:48:37 [initandlisten] options: {}  Sun Nov 06 18:48:37 [initandlisten] exception in initAndListen: 10296 dbpath (/data/db) does not exist, terminating  Sun Nov 06 18:48:37 dbexit:  Sun Nov 06 18:48:37 [initandlisten] shutdown: going to close listening sockets…  Sun Nov 06 18:48:37 [initandlisten] shutdown: going to flush diaglog…  Sun Nov 06 18:48:37 [initandlisten] shutdown: going to close sockets…  Sun Nov 06 18:48:37 [initandlisten] shutdown: waiting for fs preallocator…  Sun Nov 06 18:48:37 [initandlisten] shutdown: closing all files…  Sun Nov 06 18:48:37 [initandlisten] closeAllFiles() finished  Sun Nov 06 18:48:37 dbexit: really exiting now  E:mongobin>mongo  MongoDB shell version: 2.0.1  connecting to: test  Sun Nov 06 18:48:42 Error: couldn’t connect to server 127.0.0.1 shell/mongo.js:84  exception: connect failed  E:mongo>ls  GNU-AGPL-3.0  README  THIRD-PARTY-NOTICES  bin  data |

I was looking at

<a href="https://www.mongodb.org/display/DOCS/Quickstart+Windows"> https://www.mongodb.org/display/DOCS/Quickstart+Windows </a>

and following the instructions. Could anyone tell me what is the problem with running MongoDB (I am using Windows 7)?  
I think your log output states it clearly;

exception in initAndListen: 10296 dbpath (/data/db) does not exist, terminating  
You may simply create this directory or better to define it as a configuration value within your configuration file then use it as mongod -f C:pathtoyourmongodb.conf.  
After you’ve installed MongoDB you should firstly create a data directory for your db.  
By default MongoDB will store data in /data/db,  
but it won’t automatically create that directory. To create it, do:

|  |  |
| --- | --- |
| 1  2 | $ sudo mkdir -p /data/db/  $ sudo chown `id -u` /data/db |

You can also tell MongoDB to use a different data directory,  
with the –dbpath option.  
For more detailed information go to MongoDB wiki page.  
Specify the database path explicitly like so, and see if that resolves the issue.

|  |  |
| --- | --- |
| 1  2 | mongod –dbpath data/db  mongod –dbpath “c://data/db” |

run the above code, this will start the server.  
Check that path to database data files exists :  
Sun Nov 06 18:48:37 [initandlisten] **exception**in initAndListen: 10296 **dbpath (/data/db) does not exist,**terminating

**Q115) MongoDB: Mapreduce: reduce->multiple not supported yet**

**Ans:**I have a MongoDB collection (named “catalog”) containing about 5 astronomical catalogs. Several of these catalogs reference each other, so one of the the documents might look something like this:

{ “\_id” : ObjectId(“4ec574a68e4e7a519166015f”), “bii” : 20.9519, “class” : 2480, “cpdname” : “CPD -21 6109”, “decdeg” : -21.8417, “decpm” : 0.004, “dmname” : “-21 4299”, “hdname” : “HD 145612”, “lii” : 352.8556, “name” : “PPM 265262”, “ppmname” : “PPM 265262”, “radeg” : 243.2005, “rapm” : 0.0012, “vmag” : 9.6, “xref” : [ ] }

What I want to do is use mapreduce to move the fields such as “hdname”,”ppmname”, etc into the xref array (then unset them).  
So I try to do this one at a time, starting with the hdname field. Here are the map and reduce functions:

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13 | Skip code block  map = function() {  for (var hdname in this.hdname) {  emit(this.\_id,this.hdname);  }  }  reduce = function(key, values) {  var result = [];  for (var hdname in values) {  result.push(hdname);  }  return result;  } |

I try running the following command in the mongo shell:  
db.catalog.mapReduce(map, reduce,”catalog2?);  
Unfortunately, I get the following error:  
Thu Nov 17 15:52:17 uncaught exception: map reduce failed:

|  |  |
| --- | --- |
| 1  2  3  4  5  6 | {  “assertion” : “reduce -> multiple not supported yet”,  “assertionCode” : 10075,  “errmsg” : “db assertion failure”,  “ok” : 0  } |

Obviously I’m a newbie… can anyone help?  
Jason  
The documentation says “Currently, the return value from a reduce function cannot be an array (it’s typically an object or a number).”  
So create an object instead and wrap your array in that. Make sure also that the output of reduce is the same as the input type, so you’ll need to emit a similar value in the map operation.  
BUT … why use Map-Reduce to do this? If you emit the \_id value there’s nothing to reduce as each key will be unique. Why not just iterate over the collection copying the values and updating each record one by one?

**Q116) MongoDB sorting**

**Ans:**I want implement a “bump” feature for topics. Once a topic is bumped, it will have a new “bump\_date” field. I want to sort it so that when there is a “bump\_date” field, it will be sorted as if it was the “created” field. Here’s an example of my db.topics:

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14 | Skip code block  {  “text” : “test 1”,  “created” : “Sun Nov 20 2011 02:03:28 GMT-0800 (PST)”  },  {  “text” : “test 2”,  “created” : “Sun Nov 18 2011 02:03:28 GMT-0800 (PST)”  },  {  “text” : “test 3”,  “created” : “Sun Nov 17 2011 02:03:28 GMT-0800 (PST)”,  “bump\_date: : “Sun Nov 19 2011 02:03:28 GMT-0800 (PST)”  } |

I want the sort to return in the order of “test 1”, “test 3”, “test 2”  
Sorting in MongoDB is done like so:  
db.collection.find({ … spec … }).sort({ key: 1 })  
where 1 is ascending and -1 is descending.  
In your specific example: db.topics.find().sort({ bump\_date: 1 }), although it might be better to call it something like “updated\_at”.  
You’ll also definitely want to put an index on your “bump\_date” field.

|  |  |
| --- | --- |
| 1  2 | sorting: https://www.mongodb.org/display/DOCS/Sorting+and+Natural+Order  indexes: https://www.mongodb.org/display/DOCS/Indexes |

As Brian Hicks suggested, creating an additional updated\_at field is the way to go. This way, when a document is created you can have created\_at and updated\_at initially be the same.

|  |  |
| --- | --- |
| 1  2  3  4 | {  “created\_at”: xxx,  “updated\_at”: xxx  } |

If you then “bump” the updated\_at field by setting it to the current time when there is a a bump event you can sort on the updated\_at field to achieve the ordering you desire.

**Q117) MongoDB inserts float when trying to insert integer**

**Ans:**

|  |  |
| --- | --- |
| 1  2  3 | > db.data.update({‘name’: ‘zero’}, {‘$set’: {‘value’: 0}})  > db.data.findOne({‘name’: ‘zero})  {‘name’: ‘zero’, ‘value’: 0.0} |

How do I get Mongo to insert an integer?  
Thank you

db.data.update({‘name’: ‘zero’}, {‘$set’: {‘value’: NumberInt(0)}})

You can also use NumberLong.

**Q118) Mongodb : $in operator vs lot of single queries**

**Ans:**I know mongo is able to handle a lot of requests/sec, but let’s say I have to query a lot of documents of a collection given their \_id; what sounds better, making a $in on the \_id attribute with all the ids I want to get, or loop over findOne queries?  
Thanks by advance :)

I would definitely go with using the $in query and providing a array of \_ids instead of looping unless the documents are large and returning them all in a single query would go over the size limit of 4mb (16mb on more recent versions of MongoDB)  
Example:

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7 | db.collection.find({  “key”: { “$in”: [  ObjectId(“xxx”),  ObjectId(“yyy”),  ObjectId(“zzz”)  ]}}  ) |

Why?  
1. If you loop, there is a certain amount of setup and teardown for each query creating and exhausting cursors which would create overhead.  
2. If you are not doing this on a local machine it also creates tcp/ip overhead for every request. Locally you could use domain sockets.  
3. There is a index on “\_id” created by default and collecting a group of documents to return in a batch request should be extremely fast so there is no need to break this up into smaller queries.  
There’s some additional documentation here if you want to check it out.

**Q119) Mongodb – How to find string in multiple fields?**

**Ans:**Using Pymongo for this scenario.  
I have User that has email, first\_name, last\_name.  
I am using this Pymongo snippet:

|  |  |
| --- | --- |
| 1  2  3  4 | user\_found = users.find({‘$or’:[  {’email’:{‘$regex’:searchString, ‘$options’:’i’}},  {‘first\_name’:{‘$regex’:searchString, ‘$options’:’i’}},  {‘last\_name’:{‘$regex’:searchString, ‘$options’:’i’}}]}) |

this example works, if I want to find searchString in:  
1. email, or  
2. first\_name, or  
3. last\_name  
now I need to also find searchString in first\_name + last\_name combined.  
how can I do that? Is there a way in mongo, through the query, to combine the two into a “fullname” then search the fullname?  
Easiest way is to add an array field and populate it with all of the variants that you want to search on. Index that array field.

That way you only need one index and your search across all fields is simple and doesn’t change when you want to search on some new search variant. You can also normalize the text you put into the search array, for example, lower casing it, removing punctuation etc.

See https://stackoverflow.com/q/8206188/224370

Edit: MongoDB’s documentation now covers keyword search and the new full-text search feature.

**Q120) Get BinData UUID from Mongo as string**

**Ans:**I currently have some ids stored in Mongo as UUIDs (necessary for processing). They get returned like this:  
“\_id” : new BinData(3, “JliB6gIMRuSphAD2KmhzgQ==”)  
What would be an easy way to turn this value into a string for debugging?  
Just to be clear – the application can handle the data fine. I just need a way to get the actual UUID from Mongo quickly.

The answer to your question is more complicated that you would expect! The main reason it’s complicated is that for historical reasons (unfortunately) different drivers have written UUIDs to the database using different byte orders. You don’t mention which driver you are using, but I’ll use the C# driver as an example.  
Suppose I use the following code to insert a document:

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14 | var guid = new Guid(“00112233-4455-6677-8899-aabbccddeeff”);  collection.Insert(new BsonDocument {  { “\_id”, guid },  { “x”, 1 }  });  If I then examine the document using the Mongo shell, it looks like this:  > db.test.findOne()  { “\_id” : BinData(3,”MyIRAFVEd2aImaq7zN3u/w==”), “x” : 1 }  >  The Mongo shell has a built-in function called hex that you can use to display the binary value as a hex string:  > var doc = db.test.findOne()  > doc.\_id.hex()  33221100554477668899aabbccddeeff  > |

Look carefully: the byte order of the hex string doesn’t match the original UUID value used in the C# program. That’s because the C# driver uses the byte order returned by Microsoft’s ToByteArray method of the Guid class (which sadly returns the bytes in a bizarre order, which fact was not discovered for many months). Other drivers have their own idiosyncracies.  
To help out with this we have some helper functions written in Javascript that can be loaded into the Mongo shell. They are defined in this file:

https://github.com/mongodb/mongo-csharp-driver/blob/master/uuidhelpers.js

The Mongo shell can be told to process a file as it starts up by providing the name of the file on the command line (along with the –shell argument). Having loaded this file we have access to a number of helper functions to create and display BinData values that are UUIDs. For example:

C:mongodbmongodb-win32-x86\_64-2.0.1bin>mongo –shell uuidhelpers.js

MongoDB shell version: 2.0.1  
connecting to: test  
type “help” for help

|  |  |
| --- | --- |
| 1  2  3  4  5  6 | > var doc = db.test.findOne()  > doc.\_id.toCSUUID()  CSUUID(“00112233-4455-6677-8899-aabbccddeeff”)  > db.test.find({\_id : CSUUID(“00112233-4455-6677-8899-aabbccddeeff”)})  { “\_id” : BinData(3,”MyIRAFVEd2aImaq7zN3u/w==”), “x” : 1 }  > |

In this example the toCSUUID function is used to display a BinData value as a CSUUID and the CSUUID function is used to create a BinData value for a UUID using the C# driver’s byte ordering conventions so that we can query on a UUID. There are similar functions for the other drivers (toJUUID, toPYUUID, JUUID, PYUUID).  
Some day in the future all drivers will standardize on a new binary subtype 4 with a standard byte order. In the meantime you have to use the appropriate helper function that matches whatever driver you are using.

**Q121) All columns in MongoDB**

**Ans:**I was wondering how you would find all of the column names in a table in MongoDB, like how you use SHOW COLUMNS FROM foo; in mysql.  
MongoDB is schemaless and does not have tables. In MongoDB, each collection can have different types of items. You could store two very different items in the same collection:  
db.test.insert( { “SomeString” : “How much wood would the woodchop chop …” } );  
db.test.insert( { “Amount”: 2040.20, “Due”: new ISODate(“2012-11-10”), “UserId” : new ObjectId(“…”)} );  
usually the objects are somehow related or have a common base type, but it’s not required.  
You can, however, take a look at invidual records using  
db.collectionName.findOne()  
or  
db.collectionName.find().pretty()  
However, there’s no guarantee from MongoDB that any two records look alike or have the same fields: **there’s no schema.**

**Q122) Can’t map file memory in MongoDB**

**Ans:**The windows server 2003 box hosting our MongoDB instance ran out of disk space, and Mongo started generating the error:  
**Can’t map file memory.**  
After adding additional disk space to the server and restarting the MongoDB windows service, any operation against the database still generates the “Can’t map file memory” error. I tried doing a repair even and it gives the same error:

**> db.repairDatabase(); { “assertion” : “can’t map file memory”, “assertionCode” : 10085, “errmsg” : “db assertion failure”, “ok” : 0 }**

Any idea what I can do to get my database operational again??  
Stopping the service, deleting the lock file, and then doing a mongod –repair worked, even though db.repairDatabase did not.

The answer from Justin worked for me.  
Here are some more detailed instructions for Ubuntu:  
Stop Mongo Service: sudo service mongodb stop  
Delete Lock File: sudo rm /var/lib/mongodb/mongod.lock  
Repair the DB: sudo mongod –repair –dbpath=/var/lib/mongodb  
Restart the Mongo Service: sudo service mongodb start  
Hope that helps someone.  
(edited – note that mongodb is the service name, but mongod is the correct command for repairing)

**Q123) Why does MongoDB have collections**

**Ans:**MongoDB being document-oriented, the structure of collections seems to be a special case of documents. By that I mean one can define a document to contain other documents. So a collection is just a document containing other documents.  
So why do we need collections after all?  
Logically yes, you could design a database system like that, but practically speaking no.  
1. A collection has indexes on the documents in it.  
2. A collection requires the documents in it to have unique ids.  
3. A document is limited in size.

**Q124) How to update a field in MongoDB using existing value**

**Ans:**I’ve been looking for a way to create an update statement that will take an existing numeric field and modify it using an expression. For example, if I have a field called Price, is it possible to do an update that sets Price to 50% off the existing value ?  
So, given { Price : 19.99 }  
I’d like to do db.collection.update({tag : “refurb”}, {$set {Price : Price \* 0.50 }}, false, true);  
Can this be done or do I have to read the value back to the client, modify, then update ? I guess the question then is can expressions be used in update, and can they reference the document being updated. Thanks.

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7 | You can run server-side code with db.eval().  db.eval(function() {  db.collection.find({tag : “refurb”}).forEach(function(e) {  e.Price = e.Price \* 0.5;  db.collection.save(e);  });  }); |

Note this will block the DB, so it’s better to do find-update operation pair.

See <a href="https://www.mongodb.org/display/DOCS/Server-side+Code+Execution">https://www.mongodb.org/display/DOCS/Server-side+Code+Execution</a>

As of the release notes from upcoming Mongo 2.6 you would be able to use the new $mul operator. It would multiply the value of the field by the number with the following syntax.

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8 | {  field: { $mul: }  }  So in your case you will need to do the following:  db.collection.update(  { tag : “refurb”},  { $mul: { Price : 0.5 } }  ); |

**Q125) MongoDB $where queries and tailable cursors — WAS: date math best practices**

**Ans:**My problem: give me a list of documents older than X amount of time.  
If I have a document created with:  
db.dates.insert({date: new Date()});  
And now I want to find it only when the “date” has become 30 minutes old:  
db.dates.find({ $where: “this.date.getTime() + 30 \* 60000 <= new Date()”});  
This works, but in the Mongo documentation states quite clearly that there is a significant performance penalty to $where queries.  
Thus the question, is there a better way?

==========UPDATE 1==========  
I should have added that I am hoping to have this query function “dynamically” be creating the query one time and using it to obtain a tailable cursor on a capped collection… and I am not sure any longer that it is actually possible.  
I will test and repost.

==========UPDATE 2==========  
So, looks like my “delayed” queue is going to have to be handled in code, either with polling or some “check, then sleep” algorithm, because that appears to be what mongo’s delayed replication is doing (from db.cpp):

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15 | Skip code block  if ( replSettings.slavedelay && ( unsigned( time( 0 ) ) < nextOpTime.getSecs() + replSettings.slavedelay) ) {  assert( justOne );  oplogReader.putBack( op );  \_sleepAdviceTime = nextOpTime.getSecs() + replSettings.slavedelay + 1;  dblock lk;  if ( n > 0 ) {  syncedTo = last;  save();  }  log() << “repl:   applied ” << n << ” operations” << endl;  log() << “repl:   syncedTo: ” << syncedTo.toStringLong() << endl;  log() << “waiting until: ” << \_sleepAdviceTime << ” to continue” << endl;  return okResultCode;  } |

The $lte operator (and other range queries) will work and utilize indexes, but it cannot evaluate expressions. You have to query against a query-time constant (which would be ‘now – 30 min’):  
var threshold = new Date();  
threshold.setMinutes(-30);  
// now, query against a constant:  
db.dates.find({“date” : {$lte : threshold}});  
Of course, you can do the same with any driver for any programming language, e.g. C#

var c = db.Collection.Find(Query.LTE(“date”, DateTime.UtcNow.AddMinutes(-30));

**Q126) How to implement post tags in Mongo?**

**Ans.** I’m toying with Mongo to make a SO-like pet project, and I want to implement post tags. Each tag has a name and a slug (string to be used as an id in the URL), and a post has multiple tags. I’d like to be able to create queries like “find posts, which have tag A, don’t have tag B”, and I’m wondering what’s the mongo-way to do this.  
One way is to store an array of tag ids with each post – this will make the said query easy, but will require an extra one for each post to get tag name and a slug. Another way is to store an array of [tag name, tag slug] with each post, but I’m not sure I’ll be able to use that info in a find.  
Is there some other method, which will work better for mongo? I’m new to NoSQL, so I’d appreciate any advise on how this can be accomplished. Also, I’m using PHP binding, but that shouldn’t matter probably.  
If the tags you use and their respective slugs are unlikely to change, I think your second approach is the better one. However I would suggest a small change – rather than storing an array of [name, slug], make the fields explicit by creating a tag subdocument as in this example post document:

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16 | Skip code block  {  “\_id” : ObjectId(“4ee33229d8854784468cda7e”),  “title” : “My Post”,  “content” : “This is a post with some tags”,  “tags” : [  {  “name” : “meta”,  “slug” : “34589734”  },  {  “name” : “post”,  “slug” : “34asd97x”  },  ]  } |

You can then query for posts with a particular tag using dot notation like this:

|  |  |
| --- | --- |
| 1 | db.test.find({ “tags.name” : “meta”}) |

Because tags is an array, mongo is clever enough to match the query against any element of the array rather than the array as a whole, and dot-notation allows you to match against a particular field.  
To query for posts not containing a specific tag, use $ne:

|  |  |
| --- | --- |
| 1  2  3 | db.test.find({ “tags.name” : { $ne : “fish” }})  And to query for posts containing one tag but not the other, use $and:  db.test.find({ $and : [{ “tags.name” : { $ne : “fish”}}, {“tags.name” : “meta”}]}) |

Hope this helps!

**Q127) What operations are cheap/expensive in mongodb?**

I’m reading up on MongoDB, and trying to get a sense of where it’s best used. One question that I don’t see a clear answer to is which operations are cheap or expensive, and under what conditions.  
Can you help clarify?  
Thanks.  
It is often claimed that mongodb has insanely fast writes. While they are not slow indeed, this is quite an overstatement. Write throughput in mongodb is limited by global write lock. Yes, you heard me right, there can be only **ONE**write operation happening on the server at any given moment.  
Also I suggest you take advantage of schemaless nature of mongodb and store your data denormalized. Often it is possible to do just one disk seek to fetch all required data (because it is all in the same document). Less disk seeks – faster queries.  
If data sits in RAM – no disk seeks are required at all, data is served right from memory. So, make sure you have enough RAM.  
Map/Reduce, group, $where queries are slow.  
It is not fast to keep writing to one big document (using $push, for example). The document will outgrow its disk boundaries and will have to be copied to another place, which involves more disk operations.  
And I agree with @AurelienB, some basic principles are universal across all databases.

**Q128) MongoDB: Find a document by non-existence of a field?**

Is there a way to specify a condition of “where document doesn’t contain field” ?  
For example, I want to only find the first of these 2 because it doesn’t have the “price” field.

|  |  |
| --- | --- |
| 1  2  3 | {“fruit”:”apple”, “color”:”red”}  {“fruit”:”banana”, “color”:”yellow”, “price”:”2.00?}  db.mycollection.find( { “price” : { “$exists” : false } } ) |

<br>nostalgic URL : https://www.mongodb.org/display/DOCS/Advanced+Queries#AdvancedQueries-%24exists<br>updated URL: https://docs.mongodb.org/manual/reference/operator/query/exists/#op.\_S\_exists

**Q129) How to remove deprecated fields in Mongo?**

I have removed some fields from document definition. I want to remove this field across all documents of collection. How can I do it?  
Try:

|  |  |
| --- | --- |
| 1  2  3  4  5 | db.collection.update(  { ‘’: { ‘$exists’: true } },  // Query  { ‘$unset’: { ‘’: true  } },  // Update  false, true                          // Upsert, Multi  ) |

where field is your deprecated field and collection is the collection it was removed from.  
The general update command is of the form db.collection.update( criteria, objNew, upsert, multi ). The false and true trailing arguments disable upsert mode and enable multi update so that the query updates all of the documents in the collection (not just the first match).

**Update for MongoDB 2.2+:**

You can now provide a JSON object instead of positional arguments for upsert and multi.

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10 | db.collection.update(  { ‘’: { ‘$exists’: true } },  // Query  { ‘$unset’: { ‘’: true  } },  // Update  { ‘multi’: true }                    // Options  )  just do something like this  db.people.find().forEach(function(x) {  delete x.badField;  db.people.save(x);  }) |

oooh the $unset answer someone gave using update() here is pretty awesome too.

**Q130) Combine two $or statements**

I am trying to perform a query which is composed of two $or’s:

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12 | |——————–  | Date1  |  Date2   |  |——————–  | NULL   |  NULL    | \*  | NULL   |  TODAY   | \*  | NULL   |  TOMRW   |  | TODAY  |  TODAY   | \*  | TODAY  |  NULL    | \*  | TOMRW  |  NULL    |  |——————–  (I’ve marked the rows that would match with an asterisk)  (Date1 == null || Date1 <= today) && (Date2 == null || Date2 <= today) |

I am not sure how to express this query in MongoDB.  
It can be broken down into two individual queries that do exactly what they should:  
Skip code block

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25  26  27  28 | {  “$or”: [{  “Date1”: {  “$exists”: false  }  },  {  “Date1”: {  “$exists”: true,  “$lte”: new Date(“2012-01-07T04:45:52.057Z”)  }  }]  }  and  Skip code block  {  “$or”: [{  “Date2”: {  “$exists”: false  }  },  {  “Date2”: {  “$exists”: true,  “$lte”: new Date(“2012-01-07T04:45:52.057Z”)  }  }]  } |

Both of these select the correct set of documents – I just dont know how to execute them as a single query.  
My initial thought was to do a query like this:

|  |  |
| --- | --- |
| 1  2  3 | {  $and: [orQuery1, orQuery2]  } |

Using an $and query returns 0 results. It was explained why here in this thread: $and query returns no result  
Also in that thread, a suggestion was made to do a query like this:

|  |  |
| --- | --- |
| 1  2  3 | {  Key: {valToMatch1: 1, valToMatch2: 2}  } |

But I dont think an $or can be executed this way.  
So, the question is: How do I construct my query such that I can combine the two $or’s into a single query?  
(Its getting very late so I hope this question makes sense.)  
use test

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25  26  27  28  29  30  31  32  33  34  35  36  37  38  39  40  41  42  43  44  45  46  47  48  49  50  51  52  53 | db.test.insert({a:1})  db.test.insert({a:2, Date2:new Date(“01/07/2012”)})  db.test.insert({a:3, Date2:new Date(“01/08/2012”)})  db.test.insert({a:4, Date1:new Date(“01/07/2012”), Date2:new Date(“01/07/2012”)})  db.test.insert({a:5, Date1:new Date(“01/07/2012”)})  db.test.insert({a:6, Date1:new Date(“01/08/2012”)})  first subquery db.test.distinct(‘a’, {…});  [1, 2, 3]  second subquery db.test.distinct(‘a’, {…});  [ 1, 5, 6 ]  (Date1 == null || Date1 <= today) && (Date2 == null || Date2 <= today)  unwind  Date1 == null && Date2 == null ||  Date1 == null && Date2 <= today ||  Date1 <= today && Date2 == null ||  Date1 <= today && Date2 <= today ||  query  Skip code block  db.test.find(  {  $or :  [  {$and: [  {“Date1”: {“$exists”: false}},  {“Date2”: {“$exists”: false}}  ]},  {$and: [  {“Date1”: {“$exists”: false}},  {“Date2”: {  “$exists”: true,  “$lte”: new Date(“2012-01-07T04:45:52.057Z”)}  }  ]},  {$and: [  {“Date2”: {“$exists”: false}},  {“Date1”: {  “$exists”: true,  “$lte”: new Date(“2012-01-07T04:45:52.057Z”)}  }  ]},  {$and: [  {“Date2”: {  “$exists”: true,  “$lte”: new Date(“2012-01-07T04:45:52.057Z”)}  },  {“Date1”: {  “$exists”: true,  “$lte”: new Date(“2012-01-07T04:45:52.057Z”)}  }  ]}  ]  })  >[ 1 ] |

this should work too (assume that ‘not exist’ and ‘null’ is the same)

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16 | Skip code block  db.test.find(  {  $and :  [  {$or: [  {“Date1”: null},  {“Date1”: { “$lte”: new Date(“2012-01-07T04:45:52.057Z”)} }  ]},  {$or: [  {“Date2”: null},  {“Date2”: { “$lte”: new Date(“2012-01-07T04:45:52.057Z”)} }  ]}  ]  }  ) |

**Q131) mongodb: how to do backup of mongodb**

**Ans.** I think someone has already suggested:  
1. stop the mongod  
2. backup the data directory  
Is it reliable, I mean, ensure 100% success to restore? And I can’t find which directory stores the data… any command can help me to find it?  
If mongod process exits cleanly (that is, no crashes or kill -9 stuff), then it is safe to copy data files somewhere.  
If your current installation breaks (for example, data corruption due to unclean shutdown), you can delete its files, copy that backup over and start mongod again.  
Default data directory is /data/db, but it may be set to another value in your config file. For example, I set it to /var/lib/mongodb.  
You can also use mongodump to do a backup from a live server (this may impact performance). Usemongorestore to restore backups made by mongodump.  
At IGN we do hot backups via mongodump running as an hourly cron job, and take filer snapshots (NetApp storage) on a half-hourly basis. If your system is not too write heavy and you can afford write blocks, try using fsync and lock to flush the writes to the disk and prevent further writes. This can be followed by mongodump and upon completion you can unlock the db. Please note that you’ve to be admin to do so.  
db.runCommand({fsync:1,lock:1})

**Q132) How to update date field in mongo console?**

**Ans.** For example I want to update all records to

‘2012-01-01’ ( “time” : ISODate(“2011-12-31T13:52:40Z”) ).

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10 | db.test.update( { time : ‘2012-01-01’ }, false, true  )  return error:  Assert failed : need an object  Error(“Printing Stack Trace”)@:0  ()@shell/utils.js:35  (“assert failed : need an object”)@shell/utils.js:46  (false,”need an object”)@shell/utils.js:54  ([object Object],false,true)@shell/collection.js:189  @(shell):1  Wed Jan 11 17:52:35 uncaught exception: assert failed : need an object |

You need to create a new ISODate object like this:

db.test.insert({“Time” : new ISODate(“2012-01-10”) });

This is true both for updates and for queries. Note that your query syntax is incorrect, it should be

|  |  |
| --- | --- |
| 1  2  3  4  5 | db.test.update({ criteria }, { newObj }, upsert, multi);  For example, to update all objects, consider  db.test.update( {}, { $set : { “time” : new ISODate(“2012-01-11T03:34:54Z”) } }, true, true);  Also note that this is very different from  db.test.update( {}, { “time” : new ISODate(“2012-01-11T03:34:54Z”) }, true, false); |

because the latter will **replace**the object, rather than add a new field to the existing document or updating the existing field. In this example, I changed the last parameter to false, because multi updates only work with $ operators.

**Q133) return query based on date**

**Ans.** I have a data like this in mongodb

{ “latitude” : “”, “longitude” : “”, “course” : “”, “battery” : “0”, “imei” : “0”, “altitude” : “F:3.82V”, “mcc” : “07”, “mnc” : “007B”, “lac” : “2A83”, “\_id” : ObjectId(“4f0eb2c406ab6a9d4d000003”), “createdAt” : ISODate(“2012-01-12T20:15:31Z”) }

How to query db.gpsdatas.find({‘createdAt’: ??what here??}) so that it return me the above result from db?  
You probably want to make a range query, for example, all items created after a given date:

|  |  |
| --- | --- |
| 1 | db.gpsdatas.find({“createdAt” : { $gte : new ISODate(“2012-01-12T20:15:31Z”) }}); |

I’m using $gte (greater than or equals), because this is often used for date-only queries, where the time component is 00:00:00.  
If you really want to find a date that equals another date, the syntax would be

|  |  |
| --- | --- |
| 1 | db.gpsdatas.find({“createdAt” : new ISODate(“2012-01-12T20:15:31Z”) }); |

**Q134) MongoDB query comparing 2 fields in same collection without $where**

**Ans.** Does MongoDB supports comparing two fields in same collection by using native operators (not $where and JavaScript)? I already looked at similar questions and all answers used $where / JavaScript.  
MongoDB documentation clearly states that:  
JavaScript executes more slowly than the native operators listed on this page, but is very flexible.  
My primary concern is speed and I would like to use indexes if possible. So is comparing two fields in MongoDB possible without using JavaScript?  
This is not currently possible, but it will be possible through the new aggregation framework currently under development (2.1+). This aggregation framework is native and does not rely on relatively slow JavaScript execution paths.  
For more details check

https://www.mongodb.org/display/DOCS/Aggregation+Framework and the progress at https://jira.mongodb.org/browse/SERVER-447

**Q135) Upserts in mongodb when using custom \_id values**

**Ans.** I need to insert a document if it doesn’t exist. I know that the “upsert” option can do that, but I have some particular needs.  
First I need to create the document with its \_id field only, but only if it doesn’t exist already. My \_id field is a number generated by me (not an ObjectId). If I use the “upsert” option then I get “Mod on \_id not allowed”

|  |  |
| --- | --- |
| 1 | db.mycollection.update({ \_id: id }, { \_id: id }, { upsert: true }); |

I know that we can’t use the \_id in a $set.  
So, my question is: If there any way to a “create if doesn’t exists” atomically in mongodb?  
EDIT: As proposed by @Barrie this works (using nodejs and mongoose):

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8 | var newUser = new User({ \_id: id });  newUser.save(function (err) {  if (err && err.code === 11000) {  console.log(‘If duplicate key the user already exists’, newTwitterUser);  return;  }  console.log(‘New user or err’, newTwitterUser);  }); |

But I still wonder if it is the best way to do it.  
You can just use insert(). If the document with the \_id you specify already exists, the insert() will fail, nothing will be modified – so “create if it doesn’t exist” is what it’s already doing by default when you use insert() with a user-created \_id.  
I had the same problem, but found a better solution for my needs. You can use that same query style if you simply remove the \_id attribute from the update object. So if at first you get an error with this:

|  |  |
| --- | --- |
| 1 | db.mycollection.update({ \_id: id }, {$set: { \_id: id, name: ‘name’ }}, { upsert: true }); |

instead use this:

|  |  |
| --- | --- |
| 1 | db.mycollection.update({ \_id: id }, {$set: { name: ‘name’ }}, { upsert: true }); |

This is better because it works for both insert and update.

**Q136) How to list all collections in the mongo shell?**

**Ans.** In the MongoDB shell, how do I list all collections for the current database that I’m using? I can’t seem to find it anywhere in the docs.  
You can do:

|  |  |
| --- | --- |
| 1  2 | db.getCollectionNames()  > show collections |

will list all the collections in the currently selected DB, as stated in the command line help (help).  
> show tables  
It gives the same result as Cameron’s answer.  
Apart from the options suggested by other people:  
show collections  //output every collection  
show tables  
db.getCollectionNames() //shows all collections as a list  
There is also another way which can be really handy if you want to know how each of the collections was created (for example it is a capped collection with a particular size)  
db.system.namespaces.find()  
To list all databases  
show dbs  
enters or uses given database  
use databasename  
To list all collections:  
show collections  
Output:

|  |  |
| --- | --- |
| 1  2  3 | collection1  collection2  system.indexes |

(or)  
s

|  |  |
| --- | --- |
| 1  2  3  4  5 | how tables  //output  collection1  collection2  system.indexes |

(or)

|  |  |
| --- | --- |
| 1  2  3 | db.getCollectionNames()  //output  [ “collection1”, “collection2”, “system.indexes” ] |

To enter or use given collection  
use collectionname

**Q137) How to remove one document by Id using the official CSharp driver for Mongo**

**Ans**. can someone please show me if there is a better way to remove one document from a MongoDB using the standard 10gen driver for C# than what I have below:  
var query = Query.EQ(“\_id”, a.Id);  
database.GetCollection(“Animal”).Remove(query);  
The above works, but seems dirty and like too much work to me. The “Save” command for instance takes an instance and updates it. Perhaps I’m too used to things like EF/LINQToSQL/NHib where I’d say something like Remove(item).  
Am I missing something?  
(also, I’m trying to use the official driver rather than NoRM which seems to be dead or Samus which seems equally out of date.)  
That’s the way you do it. I’m sure you know this, but if you want to put it on one line you could combine it so you don’t need to define a query variable:  
collection.Remove(Query.EQ(“\_id”, a.Id));  
If the [id] is string, you must use ObjectId instance explicitly.  
var query = Query.EQ(“\_id”, ObjectId.Parse(id));

**Q138) Clone a collection in MongoDB**

**Ans.** I want to clone a MongoDB collection and save it on the same server with a different name. So for example right now I have the following collections: demo1.categories, demo1.users and demo2.users.  
I want to have a “demo2.categories” which is identical to “demo1.categories”. (It just has a different name.)  
Yet again the MongoDB documentation comes to the rescue  
assuming that the collection actually is named “demo1.categories”:

|  |  |
| --- | --- |
| 1 | db.demo1.categories.find().forEach( function(x){db.demo2.categories.insert(x)} ); |

The most simple & efficient way is by using copyTo(), so you can use:  
db.source.copyTo(“target”);  
& if “target” doesn’t exist, it will be created  
**— Update —**  
According to CopyTo Documentation, Because copyTo() uses eval internally, the copy operations will block all other operations on the mongod instance. So it shouldn’t be used on production environment.

**Q139) File write operations in mongo script?**

**Ans.** Is this possible to write the query result to the file from mongo js script. I have searched a lot, but I didn’t find any solution.  
ex:-

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11 | cursor = db.users.find();  while(cursor.hasNext()) {  cursor.next();  // writing the cursor output to file ????  }  You could use print and then redirect output:  script.js:  cursor = db.users.find();  while(cursor.hasNext()){  printjson(cursor.next());  } |

then run the script and redirect output to a file:

mongo –quiet script.js > result.txt

https://www.mongodb.org/display/DOCS/Scripting+the+shell

paragraph “Differences between scripted and interactive/ Printing”.

|  |  |
| --- | --- |
| 1 | ./mongo server.com/mydb –quiet –eval “db.users.find().forEach(printjson);” > 1.txt |

you can skip the while loop using forEach :

|  |  |
| --- | --- |
| 1 | db.users.find().forEach( function(user){ printjson(user) }); |

P.S: I would haved added this answer as a comment to @milan’s answer but don’t have enough reputation yet to comment :p

**Q140) Do arrays stored in MongoDB keep their order?**

**Ans.** Simple question, do arrays keep their order when stored in MongoDB?  
yep MongoDB keeps the order of the array.. just like Javascript engines..  
Yes, in fact from a quick google search on the subject, it seems that it’s rather difficult to re-order them:

https://groups.google.com/group/mongodb-user/browse\_thread/thread/1df1654889e664c1

**Q141) MongoDB: update every document on one field**

I have a collected named foo hypothetically.  
Each instance of foo has a field called lastLookedAt which is a UNIX timestamp since epoch. I’d like to be able to go through the MongoDB client and set that timestamp for all existing documents (about 20,000 of them) to the current timestamp.  
What’s the best way of handling this?  
In the Mongo shell, or with any Mongodb client, you can do something like that:  
**For Mongodb < 2.2:**

|  |  |
| --- | --- |
| 1 | db.foo.update({}, {$set: {lastLookedAt: Date.now() / 1000}}, false, true) |

<br><strong>For Mongodb >= 2.2:</strong>

|  |  |
| --- | --- |
| 1 | db.foo.update({}, {$set: {lastLookedAt: Date.now() / 1000}}, { multi: true }) |

<br>See https://www.mongodb.org/display/DOCS/Updating

1. {}is the condition (the empty condition matches any document)  
2. {$set: {lastLookedAt: Date.now()}}is what you want to do  
3. falseis for the “upsert” parameter (insert if not present, or else update – not what you want)  
4. true is for the “multi” parameter (update multiple records)

**Q142) How to check if Mongodb is properly installed**

**Ans.** I installed MongoDb yesterday on a Mac Snow Leopard and got the following error message  
Mongo::ConnectionFailure: Failed to connect to a master node at localhost:27017  
when trying to run some tests in Rails that used a mongodb.  
Another SO question mongo – ruby connection problem about the same error message had an answer that recommended removing the lock file  
sudo rm /var/lib/mongodb/mongod.lock  
but when I run that command i’m getting  
No such file or directory  
Any ideas how I can figure out how to get it working or see if it’s properly installed?  
The easiest way to run mongodb on Mac OS is:  
Download binary package from

https://www.mongodb.org/downloads,

for me, I am using lastest 64 bit version

(https://fastdl.mongodb.org/osx/mongodb-osx-x86\_64-2.0.2.tgz<br><br>1. mkdir -p $HOME/opt<br>2. cd $HOME/opt<br>3. wget https://fastdl.mongodb.org/osx/mongodb-osx-x86\_64-2.0.2.tgzto download the latest (2.0.2 for now) 64 bit binary package for Mac OS<br>4. tar xf mongodb-osx-x86\_64-2.0.2.tgz -C $HOME/optto unpack the package, and it will be unpacked to $HOME/opt/mongodb-osx-x86\_64-2.0.2<br>5. mkdir -p $HOME/opt/mongodatato create the data directory for mongodb<br>6. $HOME/opt/mongodb-osx-x86\_64-2.0.2/bin/mongod –dbpath=$HOME/opt/mongodata –7. logpath=$HOME/opt/mongod.logto start the mongodb daemon

7. Then you can run $HOME/opt/mongodb-osx-x86\_64-2.0.2/bin/mongo to connect to your local mongodb service

You can also have https://www.mongodb.org/display/DOCS/Quickstart+OS+X as additional reference

**Q143) In MongoDB, how to filter by location, sort by another field and paginate the results correctly?**

When i don’t use pagination, everything works fine (i have only 3 records in this collection, so all of them are listed here):

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10 | db.suppliers.find({location: {$near: [-23.5968323, -46.6782386]}},{name:1,badge:1}).sort({badge:-1})  { “\_id” : ObjectId(“4f33ff549112b9b84f000070”), “badge” : 3, “name” : “Dedetizadora Alvorada” }  { “\_id” : ObjectId(“4f33ff019112b9b84f00005b”), “badge” : 2, “name” : “Sampex Desentupidora e Dedetizadora” }  { “\_id” : ObjectId(“4f33feae9112b9b84f000046”), “badge” : 1, “name” : “Higitec Desentupimento e Dedetizao” }  But when i try to paginate from the first to the second page, one record doesn’t show up and one is repeated:  db.suppliers.find({location: {$near: [-23.5968323, -46.6782386]}},{name:1,badge:1}).sort({badge:-1}).skip(0).limit(2)  { “\_id” : ObjectId(“4f33ff549112b9b84f000070”), “badge” : 3, “name” : “Dedetizadora Alvorada” }  { “\_id” : ObjectId(“4f33feae9112b9b84f000046”), “badge” : 1, “name” : “Higitec Desentupimento e Dedetizao” }  db.suppliers.find({location: {$near: [-23.5968323, -46.6782386]}},{name:1,badge:1}).sort({badge:-1}).skip(2).limit(2)  { “\_id” : ObjectId(“4f33feae9112b9b84f000046”), “badge” : 1, “name” : “Higitec Desentupimento e Dedetizao” } |

Am i doing something wrong or is this some kind of bug?  
**edit**:  
Here is a workaround for this. Basically you shouldn’t mix $near queries with sorting; use $within instead.  
There is an open issue regarding the same problem. Please have a look & vote Geospatial result paging fails when sorting with additional keys

**Q144) Add a new field to a collection with value of an existing field**

I’d like to add a new field to a collection, with the value of the new field set to the value of an existing field.  
Specifically, I’d like to go from this:

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22 | # db.foo.findOne()  {  “\_id”     : ObjectId(“4f25c828eb60261eab000000”),  “created” : ISODate(“2012-01-29T16:28:56.232Z”),  “…”     : …  }  to this:  # db.foo.findOne()  {  “\_id”      : ObjectId(“4f25c828eb60261eab000000”),  “created”  : ISODate(“2012-01-29T16:28:56.232Z”),  “event\_ts” : ISODate(“2012-01-29T16:28:56.232Z”),  #same as created  “…”      : …  }  (New documents in this collection won’t all have this peculiar redundancy, but I want to do this for my existing documents)  function addEventTsField(){  db.foo.find().forEach(function(doc){  db.foo.update({\_id:doc.\_id}, {$set:{“event\_ts”:doc.created}});  });  }  Run from console:  addEventTsField(); |

**Q145) Mongodb update the specific element from subarray**

I have a collection with a following schema:

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25 | Skip code block  {  “\_id” : 28,  “n” : [{  “a” : ObjectId(“4ef8466e46b3b8140e000000”),  “c” : 28,  “p” : [ObjectId(“4f00640646b3b88005000003”), ObjectId(“4f00640146b3b88005000002”), ObjectId(“4f00637d46b3b8cc0e000001”), ObjectId(“4f00638046b3b8cc0e000002”), ObjectId(“4f00638246b3b8cc0e000003”), ObjectId(“4f00631646b3b85002000001”), ObjectId(“4f00631846b3b85002000002”)],  “u” : 26  }, {  “a” : ObjectId(“4ef8466e46b3b8140e000000”),  “c” : 10,  “p” : [ObjectId(“4f00640146b3b88005000002”), ObjectId(“4f0063fd46b3b88005000001”)],  “u” : 26  }, {  “a” : ObjectId(“4ef8467846b3b8780d000001”),  “u” : 26,  “p” : [ObjectId(“4f00637b46b3b8cc0e000000”)],  “c” : 28  }, {  “a” : ObjectId(“4ef85a3e46b3b84408000000”),  “u” : 26,  “p” : [ObjectId(“4f00631046b3b85002000000”)],  “c” : 28  }]  } |

I need to update one of the elements in the array in the document with \_id = 28 but only if the a = to some value and c = some value

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14 | Skip code block  db.coll.update({  ‘\_id’ : 28,  ‘n.a’ : new ObjectId(‘4ef85a3e46b3b84408000000’),  ‘n.c’ : 28  },  {  $push : {  ‘n.$.p’ : ObjectId(“4b97e62bf1d8c7152c9ccb74”)  },  $set : {  ‘n.$.t’ : ISODate(“2013-05-13T14:22:46.777Z”)  }  }) |

So basically I want to update specific element from array: and as far as one can see, this is the fourth element. The problem is that when the query is executing, it most likely updates the first element.  
How can I fix it?  
The problem in your code is dot-notation because When you specify the dot notation you assume that the filter criterias specified must match the single array element that satisfies all the criteria. But it doesnt. Dot notation on arrays may pickup any array element if any single criteria matches. Thats why you are getting the unexpected update.  
You have to use $elemMatch to match all the filters in the array element.

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25  26  27  28 | Skip code block  db.coll.update({  ‘\_id’ : 28,  n: {  $elemMatch:{  a : new ObjectId(‘4ef85a3e46b3b84408000000’),  c : 28 }  }  },  {  $push : {  ‘n.$.p’ : ObjectId(“4b97e62bf1d8c7152c9ccb74”)  },  $set : {  ‘n.$.t’ : ISODate(“2013-05-13T14:22:46.777Z”)  }  })  and the output is  {  “a” : ObjectId(“4ef85a3e46b3b84408000000”),  “c” : 28,  “p” : [  ObjectId(“4f00631046b3b85002000000”),  ObjectId(“4b97e62bf1d8c7152c9ccb74”)  ],  “t” : ISODate(“2013-05-13T14:22:46.777Z”),  “u” : 26  } |

**Q146) How can I rename a field for all documents in MongoDB?**

**Ans.** Assuming I have a collection in MongoDB with 5000 records, each containing something similar to:

|  |  |
| --- | --- |
| 1  2  3  4  5  6 | {  “occupation”:”Doctor”,  “name”: {  “first”:”Jimmy”,  “additional”:”Smith”  } |

Is there an easy way to rename the field “additional” to “last” in all documents? I saw the $rename operator in the documentation but I’m not really clear on how to specify a subfield.  
Not tested, but you can try:

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9 | db.foo.update({}, {$rename:{“name.additional”:”name.last”}}, false, true);  If this does not work, maybe you have to use the former way:  remap = function (x) {  if (x.additional){  db.foo.update({\_id:x.\_id}, {$set:{“name.last”:x.name.additional}, $unset:{“name.additional”:1}});  }  }  db.foo.find().forEach(remap);  please try db.collectionName.update({}, { $rename : { ‘name.additional’ : ‘name.last’ } }, { multi: true } ) |

and read this :)

https://docs.mongodb.org/manual/reference/operator/rename/#\_S\_rename

**[Upcoming Batches - MongoDB Training!](https://mindmajix.com/mongodb-training" \l "training_dates" \t "_blank)**

**[23](https://mindmajix.com/mongodb-training" \l "training_dates" \t "_blank)**

**[JUL](https://mindmajix.com/mongodb-training" \l "training_dates" \t "_blank)**

[Tuesday  
6:30 AM IST](https://mindmajix.com/mongodb-training" \l "training_dates" \t "_blank)

**[25](https://mindmajix.com/mongodb-training" \l "training_dates" \t "_blank)**

**[JUL](https://mindmajix.com/mongodb-training" \l "training_dates" \t "_blank)**

[Thursday  
6:30 AM IST](https://mindmajix.com/mongodb-training" \l "training_dates" \t "_blank)

**[27](https://mindmajix.com/mongodb-training" \l "training_dates" \t "_blank)**

**[JUL](https://mindmajix.com/mongodb-training" \l "training_dates" \t "_blank)**

[Saturday  
6:30 AM IST](https://mindmajix.com/mongodb-training" \l "training_dates" \t "_blank)

**[28](https://mindmajix.com/mongodb-training" \l "training_dates" \t "_blank)**

**[JUL](https://mindmajix.com/mongodb-training" \l "training_dates" \t "_blank)**

[Sunday  
7:00 AM IST](https://mindmajix.com/mongodb-training" \l "training_dates" \t "_blank)

[More Batches](https://mindmajix.com/mongodb-training" \l "training_dates" \t "_blank)

**Q147) Mongodb field not null delete**

**Ans.** How do I remove all documents in a collection where a field’s value isn’t null? Basically the MySql query version would be like this:  
// MySql query  
DELETE FROM companies WHERE createdBy != NULL  
// What I tried but did not work.  
$this->mongo->companies->remove(array(‘createdBy’ => true));  
I don’t even know if it is possible, if anyone could help me with this I would appreciate it  
Thanks :)  
You can do it easy via not equal operator:  
db.companies.find( { createdBy : { $ne : null } } );

**Q148) How can I see what ports mongo is listening on from mongo shell?**

**Ans.** If I have a mongo instance running, how can I check what port numbers it is listening on from the shell? I thought that db.serverStatus() would do it but I don’t see it. I see this  
“connections” : {  
“current” : 3,  
“available” : 816  
Which is close… but no. Suggestions? I’ve read the docs and can’t seem to find any command that will do this.  
From the system shell you can use lsof (see Derick’s answer below) or netstat -an to view what a process is actually doing. However, assuming you only have access to the mongo shell (which your question title implies), then you can run the serverCmdLineOpts() command. That output will give you all the arguments passed on the command line (argv) and the ones from the config file (parsed) and you can infer the ports mongod is listening based on that information. Here’s an example:  
Skip code block

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20 | db.serverCmdLineOpts()  {  “argv” : [  “./mongod”,  “-replSet”,  “test”,  “–rest”,  “–dbpath”,  “/data/test/r1”,  “–port”,  “30001”  ],  “parsed” : {  “dbpath” : “/data/test/r1”,  “port” : 30001,  “replSet” : “test”,  “rest” : true  },  “ok” : 1  } |

If you have not passed specific port options like the ones above, then the mongod will be listening on 27017 and 28017 (http console) by default. Note: there are a couple of other arguments that can alter ports without being explicit, see here:

https://docs.mongodb.org/manual/administration/security/#interfaces-and-port-numbers

You can do this from the Operating System shell by running:  
sudo lsof -iTCP -sTCP:LISTEN | grep mongo  
MongoDB only listens on one port by default (27017). If the –rest interface is active, port 28017 (27017+1000) will also be open handling web requests for details.  
MongoDB supports a getParameter command, but that only works if you’re already connected to the Database (at which point you already know the port).

**Q149) mongodb indexing embedded fields (dot notation)**

**Ans.** Let’s assume this is a document representing a customer.

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7 | {  company\_name: ‘corporate ltd.’,  pocs: [  {name: ‘Paul’, email: ‘paul@corporate.com’},  {name: ‘Jessica’, email: ‘jessica@corporate.com’}  ]  } |

I wanted to define a unique index for pocs.email So I issued the following command:

db.things.ensureIndex({“pocs.email”: 1}, {unique: true})

The strange thing is that when trying to add another company with a poc having an email already exists in another company, mongo rejects that, respecting the unique index constraint.  
that is, the following cannot exists:

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23 | Skip code block  {  company\_name: ‘corporate ltd.’,  pocs: [  {name: ‘Paul’, email: ‘paul@corporate.com’},  {name: ‘Jessica’, email: ‘jessica@corporate.com’}  ]  },  {  company\_name: ‘contoso llc’,  pocs: [  {name: ‘Paul’, email: ‘paul@corporate.com’},  ]  }  Which is fine. However, having duplicate poc within the same doc is possible, e.g.  {  company\_name: ‘corporate ltd.’,  pocs: [  {name: ‘Paul’, email: ‘paul@corporate.com’},  {name: ‘Paul’, email: ‘paul@corporate.com’},  {name: ‘Jessica’, email: ‘jessica@corporate.com’}  ]  }, |

see my cli commands sequence below:  
Skip code block

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22 | > version()  version: 2.0.2  >  > use test  switched to db test  > db.test.ensureIndex({“poc.email”: 1}, {unique: true})  >  > db.test.insert({company: “contoso”, poc: [{email: ‘me@comapny.com’}]})  > db.test.insert({company: “contoso”, poc: [{email: ‘me@comapny.com’}]})  E11000 duplicate key error index: test.test.$poc.email\_1  dup key: { : “me@comapny.com” }  > ({company: “contoso”, poc: [{email: ‘me.too@comapny.com’}, {email: ‘me.too@company.com’}]})  >  >  > db.test.find()  { “\_id” : ObjectId(“4f44949685926af0ecf9295d”), “company” : “contoso”, “poc” : [ { “email” : “me@comapny.com” } ] }  { “\_id” : ObjectId(“4f4494b885926af0ecf9295f”), “company” : “contoso”, “poc” : [ { “email” : “me.too@comapny.com” }, { “email” : “me.too@company.com” } ] }  Moreover, this happens either at insert or at update.  > db.test.update({“\_id” : ObjectId(“4f44949685926af0ecf9295d”)}, {$push: { poc: {email: ‘me@company.com’}}})  > db.test.find()  { “\_id” : ObjectId(“4f4494b885926af0ecf9295f”), “company” : “contoso”, “poc” : [ { “email” : “me.too@comapny.com” }, { “email” : “me.too@company.com” } ] }  { “\_id” : ObjectId(“4f44949685926af0ecf9295d”), “company” : “contoso”, “poc” : [        {      “email” : “me@comapny.com” },   {       “email” : “me@company.com” },   {      “email” : “me@company.com” } ] }  > |

Is this a **bug**or a **by-design-feature** I missed spotting in the documentation?  
There is an open issue regarding the same problem unique indexes not enforced within array of single document . You can vote for it.  
Also there is a nice workaround suggested by Kyle Banker in this similar post Unique indexes on embedded documents  
**Update**  
This is not only related to the embedded fields, we can reproduce the same for array fields too.

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14 | >db.uniqqueTest.insert({a:[1],x:1})  >db.uniqqueTest.createIndex({a:1}, {unique: true})  > db.uniqqueTest.find()  { “\_id” : ObjectId(“4f44c6252434860b44986b02”), “a” : [ 1 ],”x”:1 }  and it throws an error if we try to create a new document with the same value (correct behavior )  > db.uniqqueTest.insert({a:[1],x:3})  E11000 duplicate key error index: stack.uniqqueTest.$a\_1  dup key: { : 1.0 }  But this works fine if we put the same value inside the array (no errors, silently accepts the duplicate value inside the array)  > db.uniqqueTest.insert({a:[2],x:2})  > db.uniqqueTest.update({x:2},{$push:{a:2}})  { “\_id” : ObjectId(“4f44c65f2434860b44986b05”), “a” : [ 2, 2 ], “x” : 2 }  But not for this  > db.uniqqueTest.update({x:2},{$push:{a:1}])  E11000 duplicate key error index: stack.uniqqueTest.$a\_1  dup key: { : 1.0 } |

**Q150) Updating a sub-document in mongodb?**

How do I target an subdocument in the authors array as shown below, in order to update it?

collection.update({‘\_id’: “4f44af6a024342300e000001”}, {$set: { ‘authors.?’ }} )<br><br><br>The document<span class="token punctuation">:</span>

<span class="token punctuation">{</span>

\_id<span class="token punctuation">:</span> “4f44af6a024342300e000001”<span class="token punctuation">,</span>

title<span class="token punctuation">:</span> “A book”<span class="token punctuation">,</span>

created<span class="token punctuation">:</span> “<span class="token number">2012</span><span class="token operator">-</span><span class="token number">02</span><span class="token operator">-</span>22T14<span class="token punctuation">:</span><span class="token number">12</span><span class="token punctuation">:</span><span class="token number">51</span><span class="token punctuation">.</span>305Z”

authors<span class="token punctuation">:</span> <span class="token punctuation">[</span><span class="token punctuation">{</span>“\_id”<span class="token punctuation">:</span>”4f44af6a024342300e000002<span class="token operator">?</span><span class="token punctuation">}</span><span class="token punctuation">]</span>

<span class="token punctuation">}</span>

By specifying actual position <span class="token keyword">of</span> embedded document like <span class="token keyword">this</span><span class="token punctuation">:</span>

<span class="token comment" spellcheck="true">// update \_id field of first author</span>

collection<span class="token punctuation">.</span><span class="token function">update</span><span class="token punctuation">(</span><span class="token punctuation">{</span>‘\_id’<span class="token punctuation">:</span> “4f44af6a024342300e000001”<span class="token punctuation">}</span><span class="token punctuation">,</span>

<span class="token punctuation">{</span>$<span class="token keyword">set</span><span class="token punctuation">:</span> <span class="token punctuation">{</span> ‘authors<span class="token number">.0</span><span class="token punctuation">.</span>\_id’<span class="token punctuation">:</span> “<span class="token number">1</span>” <span class="token punctuation">}</span><span class="token punctuation">}</span> <span class="token punctuation">)</span>

Or via positional operator<span class="token punctuation">:</span>

<span class="token comment" spellcheck="true">// update \_id field of first matched by \_id author</span>

collection<span class="token punctuation">.</span><span class="token function">update</span><span class="token punctuation">(</span><span class="token punctuation">{</span>‘\_id’<span class="token punctuation">:</span> “4f44af6a024342300e000001”<span class="token punctuation">,</span>

<span class="token comment" spellcheck="true">//you should specify query for embedded document</span>

‘authors<span class="token punctuation">.</span>\_id’ <span class="token punctuation">:</span> “4f44af6a024342300e000002” <span class="token punctuation">}</span><span class="token punctuation">,</span>

<span class="token comment" spellcheck="true">// you can update only one nested document matched by query</span>

<span class="token punctuation">{</span>$<span class="token keyword">set</span><span class="token punctuation">:</span> <span class="token punctuation">{</span> ‘authors<span class="token punctuation">.</span>$<span class="token punctuation">.</span>\_id’<span class="token punctuation">:</span> “<span class="token number">1</span>” <span class="token punctuation">}</span><span class="token punctuation">}</span> <span class="token punctuation">)</span>

**Q151) Update MongoDB collection using $toLower**

I have an existing MongoDB collection containing user names. The user names contain both lower case and upper case letters. I want to update all the user names so they only contain lower case letters. I have tried this script, but it didn’t work

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15 | db.myCollection.find().forEach(  function(e) {  e.UserName = $toLower(e.UserName);  db.myCollection.save(e);  }  )  Any information on getting this to work will be really appreciated, Scott  MongoDB does not have a concept of $toLower as a command. The solution is to run a big forloop over the data and issue the updates individually.  You can this in any driver or from the shell:  db.myCollection.find().forEach(  function(e) {  e.UserName = e.UserName.toLowerCase();  db.myCollection.save(e);  }  ) |

You can also replace the save with an atomic update:  
db.myCollection.update({\_id: e.\_id}, {$set: {UserName: e.UserName.toLowerCase() } })  
Again, you could also do this from any of the drivers, the code will be very similar.  
EDIT: Remon brings up a good point. The $toLower command does exist as part of the aggregation framework, but this has nothing to do with updating. The documentation for updating is here.

**Q152) Mongo ObjectIDs: Safe to use in the wild?**

I’m designing an API that interacts with MongoDB.  
Now the question is, if it is **safe**to use the raw ObjectID to query for objects etc. Could any security issues arise when using the OIDs directly (e.g. in queries), or should I encrypt/decrypt them before they leave my server environment?  
Look at BSON Object ID specification here and you will know**if it is safe for you to use.**  
If you try to protect from users sending different URLs from scripts (fuskators) then it seems for me it has weak security. There won’t be too many ‘machine’, ‘pid’ part combinations. ‘time’ part can calculated if attacker can have an idea how data was inserted (especially if using batch). ‘inc’ – very weak.  
I won’t trust ObjectIDs as the only security.  
Please note there can’t be a right answer to the question “is it safe” in general. You must decide yourself.  
1. But keep in mind that such URL-based security will fall to dust when users will share URLs they visited. Even best your encryption won’t help.

**Q153) In MongoDB, How to toggle a boolean field in one document with atomic operation?**

Is there any way to toggle the boolean field of ONE document in MongoDB with atomic operation? Say, (In python)

cl.update({“\_id”: …}, {“$toggle”: {“field”: 1}})

Right now, I don’t think it’s possible to do this with one operation. The bitwise operators (https://www.mongodb.org/display/DOCS/Updating#Updating-%24bit) don’t have a ‘$xor’ yet although I’ve a patch for it.  
Right now the workaround I think think of is by always using ‘$inc’:

cl.update( { “\_id”: …}, { ‘$inc’ : { ‘field’ : 1 } } );

Then instead of checking for true or false, you can do check whether an item is “true”:

cl.find( { “\_id”: …, ‘field’ : { ‘$mod’ : [ 2, 1 ] } );

IE, you using the modulo operator to see whether it’s even or uneven with even being “unset”, and uneven being “set”. If you want to have the oppposite behaviour (ie, find all items that don’t have the flag set), then use  
[ 2, 0 ];

**Q154) Working with special characters in a Mongo collection**

I have a collection I’m unable to drop, I’m assuming that the “-” in its name is a special character. In MongoDB, what is the best way to escape special characters?

|  |  |
| --- | --- |
| 1  2  3  4  5 | > db.tweets.drop();  true  BUT  > db.tweets-old.drop();  ReferenceError: old is not defined (shell):1 |

I’ve tried to escape with quotes (both single and double) and a slash, but nothing works.  
The following works:

|  |  |
| --- | --- |
| 1 | db[“tweets-old”].drop(); |

It’s called the square bracket notation, which allows you to use special characters in property names.

**Q155) Add new field to all documents in a nested array**

I have a database of person documents. Each has a field named photos, which is an array of photo documents. I would like to add a new ‘reviewed’ flag to each of the photo documents and initialize it to false.

This is the query I am trying to use:

|  |  |
| --- | --- |
| 1 | db.person.update({ “\_id” : { $exists : true } }, {$set : {photos.reviewed : false} }, false, true) |

However I get the following error:  
SyntaxError: missing : after property id (shell):1  
Is this possible, and if so, what am I doing wrong in my update?  
Here is a full example of the ‘person’ document:  
Skip code block

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16 | {  “\_class” : “com.foo.Person”,  “\_id” : “2894”,  “name” : “Pixel Spacebag”,  “photos” : [  {  “\_id” : null,  “thumbUrl” : “https://site.com/a\_s.jpg”,  “fullUrl” : “https://site.com/a.jpg”  },  {  “\_id” : null,  “thumbUrl” : “https://site.com/b\_s.jpg”,  “fullUrl” : “https://site.com/b.jpg”  }]  } |

Bonus karma for anyone who can tell me a cleaner why to update “all documents” without using the query { “\_id” : { $exists : true } }  
Is this possible, and if so, what am I doing wrong in my update?  
No. In general MongoDB is only good at doing updates on top-level objects.  
The exception here is the $ positional operator. From the docs: Use this to find an array member and then manipulate it.  
However, in your case you want to modify all members in an array. So that is not what you need.  
Bonus karma for anyone who can tell me a cleaner why to update “all documents”  
Try db.coll.update(query, update, false, true), this will issue a “multi” update. That lasttrue is what makes it a multi.  
Is this possible,  
You have two options here:  
1. Write afor loop to perform the update. It will basically be a nested for loop, one to loop through the data, the other to loop through the sub-array. If you have a lot of data, you will want to write this is your driver of choice (and possibly multi-thread it).  
2. Write your code to handlereviewed as nullable. Write the data such that if it comes across a photo with reviewed undefined then it must be false. Then you can set the field appropriately and commit it back to the DB.  
Method #2 is something you should get used to. As your data grows and you add fields, it becomes difficult to “back-port” all of the old data. This is similar to the problem of issuing a schema change in SQL when you have 1B items in the DB.  
Instead just make your code resistant against the null and learn to treat it as a default.  
Again though, this is still not the solution you seek.

**Q156) How to listen for changes to a MongoDB collection?**

I’m creating a sort of background job queue system with MongoDB as the data store. How can I “listen” for inserts to a MongoDB collection before spawning workers to process the job? Do I need to poll every few seconds to see if there are any changes from last time, or is there a way my script can wait for inserts to occur? This is a PHP project that I am working on, but feel free to answer in Ruby or language agnostic.  
What you are thinking of sounds a lot like triggers. MongoDB does not have any support for triggers, however some people have “rolled their own” using some tricks. The key here is the oplog.  
When you run MongoDB in a Replica Set, all of the MongoDB actions are logged to an operations log (known as the oplog). The oplog is basically just a running list of the modifications made to the data. Replicas Sets function by listening to changes on this oplog and then applying the changes locally.  
**Does this sound familiar?**  
I cannot detail the whole process here, it is several pages of documentation, but the tools you need are available.  
First some write-ups on the oplog – Brief description – Layout of the local collection (which contains the oplog)  
You will also want to leverage tailable cursors. These will provide you with a way to listen for changes instead of polling for them. Note that replication uses tailable cursors, so this is a supported feature.  
MongoDB has what is called capped collections and tailable cursors that allows MongoDB to push data to the listeners.  
A capped collection is essentially a collection that is a fixed size and only allows insertions. Here’s what it would look like to create one:

db.createCollection(“messages”, { capped: true, size: 100000000 })

**Ruby example of using tailable cursors:**

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9 | coll = db.collection(‘my\_collection’)  cursor = Mongo::Cursor.new(coll, :tailable => true)  loop do  if doc = cursor.next\_document  puts doc  else  sleep 1  end  end |

**Additional Resources:**  
Ruby/Node.js Tutorial which walks you through creating an application that listens to inserts in a MongoDB capped collection.  
An article talking about tailable cursors in more detail.  
PHP, Ruby, Python, and Perl examples of using tailable cursors.

**Q157) MongoDB – Query on nested field with index**

I am trying to figure out how I must structure queries such that they will hit my index. I have documents structured like so:

|  |  |
| --- | --- |
| 1 | { “attributes” : { “make” : “Subaru”, “color” : “Red” } } |

With an index of: db.stuff.ensureIndex({“attributes.make”:1})  
What I’ve found is that querying using dot notation hits the index while querying with a document does not.  
Example:  
Skip code block

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20 | db.stuff.find({“attributes.make”:”Subaru”}).explain()  {  “cursor” : “BtreeCursor attributes.make\_1”,  “nscanned” : 2,  “nscannedObjects” : 2,  “n” : 2,  “millis” : 0,  “nYields” : 0,  “nChunkSkips” : 0,  “isMultiKey” : false,  “indexOnly” : false,  “indexBounds” : {  “attributes.make” : [  [  “Subaru”,  “Subaru”  ]  ]  }  } |

vs  
Skip code block

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14 | db.stuff.find({attributes:{make:”Subaru”}}).explain()  {  “cursor” : “BasicCursor”,  “nscanned” : 2,  “nscannedObjects” : 2,  “n” : 0,  “millis” : 1,  “nYields” : 0,  “nChunkSkips” : 0,  “isMultiKey” : false,  “indexOnly” : false,  “indexBounds” : {  }  } |

Is there a way to get the document style query to hit the index? The reason is that when constructing queries from my persistent objects it’s much easier to serialize them out as documents as opposed to something using dot notation.  
I’ll also add that we’re using a home grown data mapper layer built w/ Jackson. Would using something like Morphia help with properly constructing these queries?  
Did some more digging and this thread explains what’s going with the sub-document query. My problem above was that to make the sub-document based query act like the dot-notation I needed to use elemMatch.

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21 | Skip code block  db.stuff.find({“attributes”:{“$elemMatch” : {“make”:”Subaru”}}}).explain()  {  “cursor” : “BtreeCursor attributes.make\_1”,  “nscanned” : 2,  “nscannedObjects” : 2,  “n” : 0,  “millis” : 2,  “nYields” : 0,  “nChunkSkips” : 0,  “isMultiKey” : false,  “indexOnly” : false,  “indexBounds” : {  “attributes.make” : [  [  “Subaru”,  “Subaru”  ]  ]  }  } |

**Q158) Save Subset of MongoDB Collection to Another Collection**

I have a set like so

|  |  |
| --- | --- |
| 1  2  3  4  5 | {date: 20120101}  {date: 20120103}  {date: 20120104}  {date: 20120005}  {date: 20120105} |

How do I save a subset of those documents with the date ‘20120105’ to another collection?  
**i.e** db.subset.save(db.full\_set.find({date: “20120105”}));  
Here’s the shell version:

|  |  |
| --- | --- |
| 1  2  3 | db.full\_set.find({date:”20120105?}).forEach(function(doc){  db.subset.insert(doc);  }); |

There’s no direct equivalent of SQL’s insert into … select from ….  
You have to take care of it yourself. Fetch documents of interest and save them to another collection.  
You can do it in the shell, but I’d use a small external script in Ruby. Something like this:  
require ‘mongo’

|  |  |
| --- | --- |
| 1  2  3  4  5  6 | db = Mongo::Connection.new.db(‘mydb’)  source = db.collection(‘source\_collection’)  target = db.collection(‘target\_collection’)  source.find(date: “20120105”).each do |doc|  target.insert doc  end |

Actually, there is an equivalent of SQL’s insert into … select from in MongoDB. First, you convert multiple documents into an array of documents; then you insert the array into the target collection

db.subset.insert(db.full\_set.find({date:”20120105?}).toArray())

**Q159) What characters are NOT allowed in MongoDB field names?**

I figured out that of course . and SPACE aren’t allowed. Are there other forbidden characters ?  
You can use any (UTF8) character in the field name which aren’t special (contains “.”, or starts with “$”).

https://jira.mongodb.org/browse/SERVER-3229<br>https://stackoverflow.com/a/7976235/311220

It’s generally best to stick with lowercase alphanumeric with underscores though.

**Q160) Is MongoDB somehow limited to a single core?**

Perhaps I have misconfigured MongoDB somehow, but even under heavy load I don’t see it using more than one core. For example, top is currently showing:  
Skip code block

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10 | Tasks: 145 total,   1 running, 144 sleeping,   0 stopped,   0 zombie  Cpu0  :  0.0%us,  0.0%sy,  0.0%ni,100.0%id,  0.0%wa,  0.0%hi,  0.0%si,  0.0%st  Cpu1  :  0.0%us,  0.0%sy,  0.0%ni,100.0%id,  0.0%wa,  0.0%hi,  0.0%si,  0.0%st  Cpu2  :  0.0%us,  0.0%sy,  0.0%ni,100.0%id,  0.0%wa,  0.0%hi,  0.0%si,  0.0%st  Cpu3  :100.0%us,  0.0%sy,  0.0%ni,  0.0%id,  0.0%wa,  0.0%hi,  0.0%si,  0.0%st  Mem:  41182768k total, 40987476k used,   195292k free,   109956k buffers  Swap:  2097144k total,  1740288k used,   356856k free, 28437928k cached  PID USER      PR  NI  VIRT  RES  SHR S %CPU %MEM    TIME+  COMMAND  16297 mongod    15   0  521g  18g  18g S 99.8 47.2   2929:32 mongod  1 root      15   0 10372  256  224 S  0.0  0.0   0:03.39 init |

Is there something I can do to get Mongo to use the other cores more effectively? If it’s relevant, I currently have a big M/R running which seems to have put a lot of read queries in “waiting” mode.  
MongoDB can saturate all cores on a multi-core machine for read operations, but for write operations and map-reduce MongoDB can only utilize a single core per mongod process.  
The limitation for single-core MapReduce is due to the Javascript interpreter that MongoDB utilizes. This is something that is supposed to be fixed in the future but in the interim you can use Hadoop to execute the MapReduce and store the result set in your MongoDB database.  
Another option which has seen mixed results is to run a single mongod process for every core on the instance this will not increase performance for a single database unless they are configured to run in a sharded setup.

**Q161) Mongo count really slow when there are millions of records**

|  |  |
| --- | --- |
| 1  2  3  4  5  6 | //FAST  db.datasources.find().count()  12036788  //SLOW  db.datasources.find({nid:19882}).count()  10161684 |

Index on nid  
Any way to make the second query faster? (It is taking about 8 seconds)  
Count queries, indexed or otherwise, are slow due to the fact that MongoDB still has to do a full b-tree walk to find the appropriate number of documents that match your criteria. The reason for this is that the MongoDB b-tree structure is not “counted” meaning each node does not store information about the amount of elements in the node/subtree.  
The issue is reported here https://jira.mongodb.org/browse/SERVER-1752 and there is currently no workaround to improve performance other than manually maintaining a counter for that collection which obviously comes with a few downsides.  
Also note that the db.col.count() version (so no criteria) can take a big shortcut and doesn’t actually perform a query hence it’s speed. That said it does not always report the same value as a count query would that should return all elements (it won’t be in sharded environments with high write throughput for example). Up for debate whether or not that’s a bug. I think it is.  
Note that in 2.3+ a significant optimization was introduced that should (and does) improve performance of counts on indexed fields.

See: https://jira.mongodb.org/browse/SERVER-7745

**Q162) MongoDB not equal to**

I’m trying to display a query in MongoDB where a text field is not ” (blank)

|  |  |
| --- | --- |
| 1 | { ‘name’ : { $not : ” }} |

However I get the error invalid use of $not  
I’ve looked over the documentation but the examples they use are for complicated cases (with regexp and $not negating another operator).  
How would I do the simple thing I’m trying to do?  
Use $ne — $not should be followed by the standard operator:  
An examples:  
use test  
switched to db test

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7 | db.test.insert({author : ‘me’, post: “”})  db.test.insert({author : ‘you’, post: “how to query”})  db.test.find({‘post’: {$ne : “”}})  { “\_id” : ObjectId(“4f68b1a7768972d396fe2268”), “author” : “you”, “post” : “how to query” }  And now $not:  db.test.find({‘post’: {$not: {$ne : “”}}})  { “\_id” : ObjectId(“4f68b19c768972d396fe2267”), “author” : “me”, “post” : “” } |

Use $ne instead of $not

<strong>https://docs.mongodb.org/manual/reference/operator/ne/#op.\_S\_ne</strong><br>db.collections.find({“name”: {$ne: “”}});

If you want to do multiple $ne then do

|  |  |
| --- | --- |
| 1 | db.users.find({name : {$nin : [“mary”, “dick”, “jane”]}}) |

**Q163) Multiple $elemMatch expressions for matching array values using $all in MongoDB?**

In the answer to a question I found a interesting solution for searching array values using $elemMatch.  
If we have the following documents in our collection:

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8 | {  foo : [ { bar : “xy”, baz : 1 },  { bar : “a”, baz : 10 } ]  },  {  foo : [ { bar : “xy”, baz : 5 },  { bar : “b”, baz : 50 } ]  } |

The following query will match only the first document:

|  |  |
| --- | --- |
| 1  2  3 | db.test.find({  foo : { “$all” : [ { “$elemMatch” : { bar : “xy”, baz : 1} }, { “$elemMatch” : { bar : “a”, baz: 10 } } ] }  }); |

I tried it with several other examples and it really works. But the official documentation for $alloperator doesn’t say anything about combining these two queries.  
Is this the intended behavior or a bug? Or is this just a problem that the documentation does not cover this use case?  
This is the intended behavior. The documentation doesn’t cover this use case and we are working on it to make it better. Its difficult, however, to document every possible query combination.

**Q164) How to remove duplicate entries from an array?**

How to remove duplicate entries from an array?  
In below example “Algorithms in C++” is added twice.  
$unset modifier removes a particular field but how to remove an entry from a field?

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7 | > db.users.find()  { “\_id” : ObjectId(“4f6cd3c47156522f4f45b26f”),  “favorites” : { “books” : [ “Algorithms in C++”,  “The Art of Computer Programmning”,  “Graph Theory”,  “Algorithms in C++” ] },  “name” : “robert” } |

What you have to do is use map reduce to detect and count duplicate tags .. then use $set to replace the entire books based on { “\_id” : ObjectId(“4f6cd3c47156522f4f45b26f”),  
This has been discussed sevel times here .. please seee  
Removing duplicate records using MapReduce  
Fast way to find duplicates on indexed column in mongodb

https://csanz.posterous.com/look-for-duplicates-using-mongodb-mapreduce<br>https://www.mongodb.org/display/DOCS/MapReduce

How to remove duplicate record in MongoDB by MapReduce?  
Thanks :)  
As of MongoDB 2.2 you can use the aggregation framework with an $unwind, $group and$project stage to achieve this:

|  |  |
| --- | --- |
| 1  2  3  4  5  6 | db.users.aggregate([{$unwind: ‘$favorites.books’},  {$group: {\_id: ‘$\_id’,  books: {$addToSet: ‘$favorites.books’},  name: {$first: ‘$name’}}},  {$project: {‘favorites.books’: ‘$books’, name: ‘$name’}}  ]) |

Note the need for the $project to rename the favorites field, since $group aggregate fields cannot be nested.

**Q165) Order of $lt and $gt in MongoDB range query**

Today I have noticed that the order in which the $lt and $gt operators are given seem to matter in MongoDB 2.0.2.  
I have a database of games. “player” is an array of two strings representing both players, “endedAtMS” is a timestamp when the game has ended. I have created this index:  
db.games.ensureIndex({player:1,endedAtMS:-1})  
To get 30 of my games which were finished in a certain time range, ordered by the time the games where finished, I do:

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25  26  27  28  29  30  31  32 | Skip code block  db.games.find({ “player” : “Stefan” ,  “endedAtMS” : { “$lt” : 1321284969946 ,  “$gt” : 1301284969946}}).  sort({endedAtMS:-1}).  limit(30).  explain()  {  “cursor” : “BtreeCursor player\_1\_endedAtMS\_-1”,  “nscanned” : 30,  “nscannedObjects” : 30,  “n” : 30,  “millis” : 0,  “nYields” : 0,  “nChunkSkips” : 0,  “isMultiKey” : true,  “indexOnly” : false,  “indexBounds” : {  “player” : [  [  “Stefan”,  “Stefan”  ]  ],  “endedAtMS” : [  [  1321284969946,  -1.7976931348623157e+308  ]  ]  }  } |

All seems to work fine. However when I change the order of $lt and $gt in the query above I get this:

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25  26  27  28  29  30  31  32 | Skip code block  db.games.find({ “player” : “Stefan” ,  “endedAtMS” : { “$gt”:1301284969946,  “$lt” : 1321284969946}}).  sort({endedAtMS:-1}).  limit(30).  explain()  {  “cursor” : “BtreeCursor player\_1\_endedAtMS\_-1”,  “nscanned” : 126,  “nscannedObjects” : 126,  “n” : 30,  “millis” : 1,  “nYields” : 0,  “nChunkSkips” : 0,  “isMultiKey” : true,  “indexOnly” : false,  “indexBounds” : {  “player” : [  [  “Stefan”,  “Stefan”  ]  ],  “endedAtMS” : [  [  1.7976931348623157e+308,  1301284969946  ]  ]  }  } |

As you can see 126 docs need to be scanned to get the 30 docs for the result. If you take a look at the indexBounds in the explain output it seems that only the first operator is used to limit the search space in the index.  
What do I miss? Why is Mongo only using one operator to limit the search space?  
This is a known issue. The short answer is that it has to do with the fact that a multikey index is used (“player” is an array), and the index cannot be constrained on both upper and lower bounds.  
This is explained in more detail in the Jira case: https://jira.mongodb.org/browse/SERVER-4155 – “Index bound incorrect?”  
There is an open Jira ticket to improve this behavior: https://jira.mongodb.org/browse/SERVER-4180 – “Wrong indexbounds picked for a date range query (regression)” which is slated to be released in version 2.1.2 (this version is subject to change). Please vote for it!

**Q166) Foreign key like relationship in Mongo DB**

How can I implement a foreign key like relationship in Mongo DB?  
hiya see this: MongoDB normalization, foreign key and joining &&

further ===>https://books.google.com/books/about/Document\_Design\_for\_MongoDB.html?id=TbIHkgEACAAJ&redir\_esc=y

hope it helps! cheerios!  
MongoDB doesn’t support server side foreign key relationships, normalization is also discouraged. You should embed your child object within parent objects if possible, this will increase performance and make foreign keys totally unnecessary. That said it is not always possible, so there is a special construct called DBRef which allows to reference objects in a different collection. This may be then not so speedy because DB has to make additional queries to read objects but allows for kind of foreign key reference.  
Still you will have to handle your references manually. Only while looking up your DBRef you will see if it exists, the DB will not go through all the documents to look for the references and remove them if the target of the reference doesn’t exist any more. But I think removing all the references after deleting the book would require a single query per collection, no more, so not that difficult really.

**Edit update**

<br>https://levycarneiro.com/tag/mongodb/

levycarneiro.com/tag/mongodb [quote] So you create 4 collections: Clients, Suppliers, Employees and Contacts. You connect them all together via a db reference. This acts like a foreign key. But, this is not the mongoDB way to do things. Performance will penalized. [unquote]

**Q167) How to get a specific embedded document inside a MongoDB collection?**

I have a collection Notebook which has embedded array document called Notes. The sample  
document looks like as shown below.

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15 | Skip code block  {  “\_id” : ObjectId(“4f7ee46e08403d063ab0b4f9”),  “name” : “MongoDB”,  “notes” : [  {  “title” : “Hello MongoDB”,  “content” : “Hello MongoDB”  },  {  “title” : “ReplicaSet MongoDB”,  “content” : “ReplicaSet MongoDB”  }  ]  } |

I want to find out only note which has title “Hello MongoDB”. I am not getting what should  
be the query. Can anyone help me.  
I don’t believe what you are asking is possible, at least without some map-reduce maybe?  
See here: Filtering embedded documents in MongoDB  
That answer suggests you change your schema, to better suit how you’d like to work with the data.  
You can use a either “dot notation” or $elemMatch to get back the correct, document that has the matching “note title” …  
>

|  |  |
| --- | --- |
| 1  2  3 | db.collection.find({ “notes.title” : “Hello MongoDB”}, { “notes.title” : 1?});  or …  > db.collection.find({ “notes” : { “$elemMatch” : { “title” : “Hello MongoDB”} }}); |

But you will get back the whole array, not just the array element that caused the match.  
Also, something to think about … with your current setup it woud be hard to do any operations on the items in the array.  
If you don’t change your schema (as the answer linked to suggests) … I would consider adding “ids” to each element in the array so you can do things like delete it easily if needed.  
You can do this with mongo version higher 2.2  
the query like this:

|  |  |
| --- | --- |
| 1 | db.coll.find({ ‘notes.title’: ‘Hello MongoDB’ }, {‘notes.$’: 1}); |

you can try with $elemMatch like Justin Jenkins

**Q168) can you have mongo $push prepend instead of append?**

I’d like to have push add at the beginning of my set rather than appended to the back when I do a mongo $push.  
Is it possible to do an atomic push update that adds it as the first element rather than the last?  
A similar question was asked a few days ago. Unfortunately, the short answer is, “no”, but there is an open request for this feature.  
https://jira.mongodb.org/browse/SERVER-2191 – “$push() to front of array”  
There is some more information as well as a possible work-around on the other thread: “Use MongoDB array as stack” – Use MongoDB array as stack  
Hopefully the above will be useful and help you to find an acceptable work-around.  
Use negative index with $set for prepend, tested in mongo v2.2:

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23 | Skip code block  > db.test.insert({‘array’: [4, 5, 6]})  > db.test.find()  { “\_id” : ObjectId(“513ad0f8afdfe1e6736e49eb”),  “array” : [ 4, 5, 6 ] }  //prepend 3  > db.test.update({“\_id” : ObjectId(“513ad0f8afdfe1e6736e49eb”)},  {‘$set’: {‘array.-1’:     3}})  > db.test.find()  { “\_id” : ObjectId(“513ad0f8afdfe1e6736e49eb”),  “array” : [ 3, 4, 5, 6 ] }  //prepend 2  > db.test.update({“\_id” : ObjectId(“513ad0f8afdfe1e6736e49eb”)},  {‘$set’: {‘array.-1’:     2}})  > db.test.find()  { “\_id” : ObjectId(“513ad0f8afdfe1e6736e49eb”),  “array” : [ 2, 3, 4, 5, 6 ] }  //prepend 1  > db.test.update({“\_id” : ObjectId(“513ad0f8afdfe1e6736e49eb”)},  {‘$set’: {‘array.-1’:     1}})  > db.test.find()  { “\_id” : ObjectId(“513ad0f8afdfe1e6736e49eb”),  “array” : [ 1, 2, 3, 4, 5, 6 ] } |

As of MongoDB v2.5.3, there is a new $position operator that you can include along with the$each operator as part of your $push query to specify the location in the array at which you would like to insert a value.  
Here’s an example from the docs page to add the elements 20 and 30 at the array index of 2::

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8 | db.students.update( { \_id: 1 },  { $push: { scores: {  $each: [ 20, 30 ],  $position: 2  }  }  }  ) |

<br>Reference:https://docs.mongodb.org/master/reference/operator/update/position/#up.\_S\_position

**Q169) How can I create unique IDs for embedded documents in MongoDB?**

So I need to reference particular subdocuments uniquely from items in my collection. For instance:

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7 | User = {  ‘name’: ‘jim’,  ‘documents: [  {‘id’: 0001, ‘title’: “My document”},  {‘id’: 0002, ‘title’: “My second document!”},  ]  } |

So I need to be able to auto-create IDs for new documents, preferably not at the application level (since there will be race conditions in the actual development scenario).  
Is there a way to use mongo’s autogenerated ObjectId (used in the \_id field at the collection level), or something similar?  
Yes, using mongo’s ObjectId is the way to go. The only thing is: you have to generate them yourself, in the application code. They are meant to be globally unique, different workers won’t generate two identical ObjectIds, so there’s no race condition in that sense.  
All official drivers should provide a way to generate ObjectId. Here’s how it is in Ruby:

|  |  |
| --- | --- |
| 1  2  3  4  5  6 | oid = BSON::ObjectId.new  All drivers have functionality for generating ObjectIds.  In the shell you just do new ObjectId():  > db.test.insert({x:new ObjectId()});  > db.test.find();  { “\_id” : ObjectId(“4f88592a06c05e4de90d0bc1”), “x” : ObjectId(“4f88592a06c05e4de90d0bc0”) } |

In Java it’s new ObjectId() as well. See the API docs for your driver to see the specific syntax.

**Q170) Deleting a key/value from existing MongoDB entry**

I need to delete a certain key and value from every entry in a particular collection. I’ve looked into remove and that seems to be for only entire entries. Looking at update, I don’t believe updating a particular key with null or an empty string would achieve what I’m trying to do. I’m very much a beginner with mongodb, so please excuse my ignorance.  
Long story short, how can I turn

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10 | {  “\_id” : 1234,  “name” : “Chris”,  “description” : “Awesome”  }  into  {  “\_id” : 1234,  “name” : “Chris”  } |

without deleting the entry and creating a new one, or using any non-mongodb commands? Thanks!  
Try $unset in a call to update().  
In other words,

|  |  |
| --- | --- |
| 1 | db.collection\_name.update({ \_id: 1234 }, { $unset : { description : 1} }); |

should work. Link.

**Q171) Problems with Mongo value-in-array query: $not operator doesn’t work, and ObjectIds cannot be selected**

I am trying to select a document which does NOT contain a value in a document’s array.  
I’m having two problems, which I’ll present separately:

**Q172) Why does this not work?**

(2) I cannot get the value in array query to work if the array values are ObjectIds:

|  |  |
| --- | --- |
| 1  2  3  4  5  6 | { \_id: Object(“000000000000000000000000”), mylist: [ ObjectId(“111111111111111111111111”) ] }  The following does NOT retrieve this document:  myCol.find({mylist:ObjectId(“111111111111111111111111”)})  Can someone suggest what I might be doing wrong?  There is no $not that works like, you want $ne for simple cases like yours:  db.myCol.find({ mylist: { $ne: ‘orange’ } }) |

Your second should (and does) work fine for me, perhaps you’re using the wrong number of ones.

**Q173) Updating a document field in mongo based on another field’s value**

Suppose I have two fields F1 and F2. I want to update F1 to become F1 + “, ” + F2. Can I do so with a single update command in mongo?  
No, you can’t do that. You can’t use expressions in mongodb updates. The only way is to fetch this document to the client, compose new field value there and issue a prepared update statement.

**Q174) mongodb map reduce to get uniques by date/day**

trying to group by date and number

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7 | my document  { “\_id” : ObjectId(“4f956dee76ddb26752026e8f”), “request” : “default”, “options” : “1”, “keyword” : “somekey”, “number” : “5b234b79-4d70-437e-8eef-32a2941af40a”, “date” : “20120423200446”, “time” : 1335193066 }  my query  map = “function() { var date = new Date(this.time \* 1000); var key = date.getFullYear() + date.getMonth() + date.getDate(); emit({day : key, number: this.number}, {count: 1}); }”  reduce = “function(key, values) { var count = 0; values.forEach(function(v) { count += v[‘count’]; }); return {count: count}; }”  db.txtweb.mapReduce(map, reduce, {out: “pageview\_results”});  my error |

uncaught exception: map reduce failed:{ “errmsg” : “ns doesn’t exist”, “ok” : 0 }  
I cannot figure out whats wrong, but I think it is do something with the date functionality.  
Any ideas.  
ns doesn’t exist means that you’re accessing a non-existent database or collection. Check their names  
Script below group this.amount by day (without time)

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22 | Skip code block  db.payments.mapReduce(  function() {  var k = clone(this.payment\_time);  k.setMinutes(0);  k.setSeconds(0);  k.setMilliseconds(0);  emit(k, this.amount);  },  function (key, vals) {  var total = 0;  for (var i = 0; i < vals.length; i++) {  total += vals[i];  }  return total;  },  {  out : {merge : “resultName” }  }  );  }  }; |

**Q175) mongo – how to query a nested json**

I am a complete mongo newbie. I am using mongo hub for mac. I need to query the for following json –

|  |  |
| --- | --- |
| 1  2  3  4  5  6 | {“\_id” : ObjectId( “abcd” ),  “className” : “com.myUser”,  “reg” : 12345,  “test” : [  { “className” : “com.abc”,  “testid” : “pqrs” } ] } |

and find records where testid is pqrs. How would I go about doing that?  
You can type {‘test.testid’: ‘pqrs’} in the query field of Mongo Hub.

**Q176) Mongodb find a document with all subdocuments satisfying a condition**

I have Game collection in my DB:

|  |  |
| --- | --- |
| 1  2  3  4 | var game = {  players: [{username:”user1?, status:”played”},  {username:”user2?, status:”accepted”}]  } |

As far as I understand query like this: db.games.find({“players.status”:”played”}) will give me all games where at least one player has status “played”. How can I find games with ALL players having status “played”?  
If you only have one other status than “played” use the query:

|  |  |
| --- | --- |
| 1 | db.games.find({“players.status”:{$ne:”accepted”}}) |

You can adjust the query to handle more status values, as long as they are all known at the time of the query.

**Q177) MongoDB – Update an object in nested Array**

Assume we have the following collection, which I have few questions about:

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20 | Skip code block  {  “\_id” : ObjectId(“4faaba123412d654fe83hg876”),  “user\_id” : 123456,  “total” : 100,  “items” : [  {  “item\_name” : “my\_item\_one”,  “price” : 20  },  {  “item\_name” : “my\_item\_two”,  “price” : 50  },  {  “item\_name” : “my\_item\_three”,  “price” : 30  }  ]  } |

1 – I want to increase the price for “item\_name”:”my\_item\_two” **and if it doesn’t exists**, it should be appended to the “items” array.  
2 – How can I update two fields at the same time. For example, increase the price for “my\_item\_three” and at the same time increase the “total” (with the same value).  
I prefer to do this on the MongoDB side, otherwise I have to load the document in client-side (Python) and construct the updated document and replace it with the existing one in MongoDB.  
**UPDATE**This is what I have tried and works fine **IF THE Object Exists :**  
db.test\_invoice.update({user\_id : 123456 , “items.item\_name”:”my\_item\_one”} , {$inc: {“items.$.price”: 10}})  
But if the key doesn’t exist it does nothing. Also it only updates the nested object. There is no way with this command to update the “total” field as well.  
For question #1, let’s break it into two parts. First, increment any document that has “items.item\_name” equal to “my\_item\_two”. For this you’ll have to use the positional “$” operator. Something like:

|  |  |
| --- | --- |
| 1  2  3  4 | db.bar.update( {user\_id : 123456 , “items.item\_name” : “my\_item\_two” } ,  {$inc : {“items.$.price” : 1} } ,  false ,  true); |

Note that this will only increment the first matched subdocument in any array (so if you have another document in the array with “item\_name” equal to “my\_item\_two”, it won’t get incremented). But this might be what you want.  
The second part is trickier. We can push a new item to an array without a “my\_item\_two” as follows:

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9 | db.bar.update( {user\_id : 123456, “items.item\_name” : {$ne : “my\_item\_two” }} ,  {$addToSet : {“items” : {‘item\_name’ : “my\_item\_two” , ‘price’ : 1 }} } ,  false ,  true);  For your question #2, the answer is easier. To increment the total and the price of item\_three in any document that contains “my\_item\_three,” you can use the $inc operator on multiple fields at the same time. Something like:  db.bar.update( {“items.item\_name” : {$ne : “my\_item\_three” }} ,  {$inc : {total : 1 , “items.$.price” : 1} ,  false ,  true); |

**Q178) To what extent are ‘lost data’ criticisms still valid of MongoDB?**

I’m referring to the following

https://pastebin.com/raw.php?i=FD3xe6Jt

Which, if still valid would be worrying to some extent. The article primarily references v1.6 and v1.8 and since then v2 has been released. Are the shortcomings discussed in the article still outstanding as of the current release?  
That particular post was debunked, point by point by the MongoDB CTO and co-founder, Eliot Horowitz, here:

https://news.ycombinator.com/item?id=3202959

There is also a good summary here:

https://www.betabeat.com/2011/11/10/the-trolls-come-out-for-10gen/

The short version is, it looks like this was basically someone trolling for attention (successfully), with no solid evidence or corroboration. There have been genuine incidents in the past, which have been dealt with as the product evolved (see the introduction of journaling in 1.8 for example) or as more specific bugs were found and fixed.  
**Disclaimer**: I do work for MongoDB (formerly 10gen), and love the fact that philnate got here and refuted this independently first – that probably says more about the product than anything else :)  
**Update: August 19th 2013**  
I’ve seen quite a bit of activity on this answer recently, which I assume is related to the announcement of the bug in SERVER-10478 – it is most certainly an edge case, but I would still recommend anyone using sharding with large documents to upgrade ASAP to v2.2.6 and v2.4.6 which include the fix for this issue.  
Never heard of those severe problems in recent versions. What you need to consider is that MongoDB has no decade of development as relational Systems in the back. Further it may be true that MongoDB doesn’t offer that much functionality to avoid data loss at all. But even with relational Systems you won’t be ever sure that you’ll never loose any data. It highly depends on your system configuration (so with Replication and manual data backups you should be quite safe).  
As a general guideline to avoid Beta Bugs or bugs from early versions, avoid to use fresh versions in productions (there’s a reason why debian is so popular for servers). If MongoDB would suffer such severe problems (all the time) the list of users would be smaller:https://www.mongodb.org/display/DOCS/Production+Deployments Additionally I don’t really trust this pastebin message, why is this published anonymously? Is this person company shamed to tell that they used mongodb, do they fear 10gen? Where a links to those Bug reports (or did 10gen delete them from JIRA?)  
So lets talk shortly about those points:  
1. Yep MongoDB operates normally in fire and forget mode. But you can modify this bevavior with several options: https://www.mongodb.org/display/DOCS/getLastError+Command. So only because MongoDB defaults to it, it doesn’t mean you can’t change it to your needs. But you need to live less performance if you don’t fire and forget within your app, as you’re adding a roundtrip.  
2. Never heard of such problems, except those caused to own failure…but that can happen with relational systems as well. I guess this point only talks about Master-Slave Replication. Replica-Sets are much never and stable. Some links from the web where other dbms caused data loss due to malfunction as well: https://blog.lastinfirstout.net/2010/04/bit-by-bug-data-loss-running-oracle-on.html https://dbaspot.com/oracle-server/430465-parallel-cause-data-lost-another-oracle-bug.html https://bugs.mysql.com/bug.php?id=18014 (Those posted links aren’t in any favor of a given system or should imply anything else than showing that there are bugs in other systems as well, who can cause data loss.)  
3. Yes actually there’s Locking at instance level, I don’t think that in sharded environment this is a global one, I think this will be at instance level for each shard separate, as there’s no need to lock other instances as there are no consistency checks needed. The upcoming Version 2.2 will lock at DB Level, tickets for Collection Level and maybe extend or document exists as well (https://jira.mongodb.org/browse/SERVER-4328). But locking at deeper levels may affect the actual performance of MongoDB, as a lock management is expensive.  
4. Moving chunks shouldn’t cause much problems as rebalancing should take a few chunks from each node and move them to the new one. It never should cause ping/pong of chunks nor does rebalancing start just because of a difference of one or two chunks. What can be problematic is when your shard key is choosen wrong. So you may end up with all new entries inserted to one node rather than all. So you would see more often rebalancing which can cause problems, but that would be not due to mongo rather than your poorly choosen shardkey.  
5. Can’t comment on this one  
6. Not 100% sure, but I think Replicasets where introduced in 1.6, so as told earlier never use the latest version for production, except you can live with loss of data. As with every new feature there’s the possibility of bugs. Even extensive testing may not reveal all problems. Again always run some manual backup for gods sake, except you can live with data loss.  
7. Can’t comment on this. But in reality software may contain severe bugs. Many games suffer those problems as well and there are other areas as well where banana software was quite well known or is. Can’t Comment about something concrete as this was before my MongoDB time.  
8. Replication can cause such problems. Depending on the replication strategy this may be a problem and another system may fit better. But without a really really write intensive workload you may not encounter such problems. Indeed it may be problematic to have 3 replicas polling changes from one master. You could cure the problem by adding more shards.  
As a general conclusion: Yeah it may be that those problems were existent, but MongoDB did much in this direction and further I doubt that other DBMS never had the one or other problem itself. Just take traditional relational dbms, would those scale well to web-scale there would be no need for Systems like MongoDB, HBase and what else. You can’t get a system which fits all needs. So you have to live with the downsides of one or try to build a combined system of multiple to get what you need.  
Disclaimer: I’m not affiliated with MongoDB or 10gen, I’m just working with MongoDB and telling my opinion about it.  
I can’t speak for every case, only my own. However I don’t work for 10gen or its competitors, and I have lost data when using MongoDB, so here goes.  
The main criticisms of Mongo around the last time I used it (2010) were:  
1. Supposedly stable versions of Mongo had major data-losing bugs that weren’t made explicit to users. Eg, prior to 1.8 non-clustered configurations were likely to lose data. This was documented by Mongo, but not to the extent a data losing bug in a stable-versioned DB would normally be.  
The main defence of that criticism was:  
2. Users were informed of this danger, albeit not so explicitly. Users should read all the documentation before they begin.  
In my own case, I was using 1.7 in a single server configuration but aware of the risk. I shut down the DB to take a back up. The act of shutting down the DB itself lost my data, 10gen assisted (for free) but were unable to recover the data.

**Q179) Mongo DB Delete a field and value**

In mongo shell how would I delete all occurrences of “id” : “1” the value of the field is always different. Would I use the $unset operator? Would that delete the value and the field?  
You’re saying remove all occurrences of the field, right? If so, then it should be like this:

|  |  |
| --- | --- |
| 1  2  3  4  5  6 | db.collection.update(  { id: { $exists: true } },  // criteria  { $unset: { id: 1 } },      // modifier  false,                      // no need to upsert  true                        // multi-update  ); |

**Q180) Checking if a field contains a string**

I’m looking for an operator, which allows me to check, if the value of a field contains a certain string.  
Something like:

|  |  |
| --- | --- |
| 1 | db.users.findOne({$contains:{“username”:”son”}}) |

Is that possible?  
You can do it with the following code.

|  |  |
| --- | --- |
| 1 | db.users.findOne({“username” : {$regex : “.\*son.\*”}}); |

As Mongo shell support regex, that’s completely possible.

|  |  |
| --- | --- |
| 1 | db.users.findOne({“username” : /.\*son.\*/}); |

If we want the query to be case-insensitive, we can use “i” option, like shown below:

|  |  |
| --- | --- |
| 1 | db.users.findOne({“username” : /.\*son.\*/i}); |

See: https://www.mongodb.org/display/DOCS/Advanced+Queries#AdvancedQueries-RegularExpressions<br>https://www.mongodb.org/display/DOCS/SQL+to+Mongo+Mapping+Chart<br>https://php.net/manual/en/mongo.sqltomongo.php

|  |  |
| --- | --- |
| 1  2  3  4 | MySQL  SELECT \* FROM users WHERE username LIKE “%Son%”  MongoDB  db.users.find({username:/Son/}) |

**Q181) What’s the fastest way to copy a collection within the same database?**

I want to copy a collection within the **same**database and give it a different name – basically take a snapshot.  
What’s the best way to do this? Is there a command, or do I have to copy each record in turn?  
I’m aware of the cloneCollection command, but it seems to be for copying to another server only.  
I’m also aware of mongoimport and mongoexport, but as I’m doing this via PHP I’d prefer not to make calls out to the shell.  
You have a few options, but the fastest is:  
mongoexport -d db -c sourcecollection | mongoimport -d db -c targetcollection –drop  
or in php:

|  |  |
| --- | --- |
| 1 | `mongoexport -d db -c sourcecollection | mongoimport -d db -c targetcollection –drop`; |

after that you have

|  |  |
| --- | --- |
| 1 | mongo db < script.js |

where, as shown in the mongo docs, script.js contains something like:

|  |  |
| --- | --- |
| 1 | db.myoriginal.find().forEach( function(x){db.mycopy.insert(x)} ); |

The slowest (by an order of magnitude or more) way to copy a collection will be to use the native php driver – simply because of moving information around. But you could issue the above mongo query if you absolutely want to avoid cli calls using the db execute function.

**Q182) mongodb: upserting: only set value if document is being inserted**

Considering a simple mongo document structure:  
{ \_id, firstTime, lastTime }  
The client needs to insert a document with a known ID, or update an existing document. The ‘lastTime’ should always be set to some latest time. For the ‘firstTime’, if a document is being inserted, then the ‘firstTime’ should be set to current time. However, if the document is already created, then ‘firstTime’ remain unchanged. I would like to do it purely with upserts (to avoid look ups).

<br>I’ve crawled the https://www.mongodb.org/display/DOCS/Updating,

but I just don’t see how that particular operation can be done.  
I don’t believe this is something unreasonable, there are $push and $addToSet operations that effectively do that on array fields, just nothing that would do the same on simple fields. It’s like there should be something like $setIf operation.  
I ran into the exact same problem and there was no simple solution for <2.4 however since 2.4 the **$setOnInsert** operator let’s you do exactly that.

|  |  |
| --- | --- |
| 1  2  3  4 | db.collection.update( ,  { $setOnInsert: { “firstTime”: } },  { upsert: true }  ) |

See the 2.4 release notes of setOnInsert for more info.

**Q183) Change collection name in mongodb**

Changing the name of a collection in mongodb can be achieved by copy the documents in the collection to a new one and delete the original.  
But is there a simpler way to change the name of a collection in mongodb?  
db.oldname.renameCollection(“newname”)  
Really? No google search?

https://www.mongodb.org/display/DOCS/renameCollection+Command<br><br><br><span class="token operator">></span> db<span class="token punctuation">.</span>oldname<span class="token punctuation">.</span><span class="token function">renameCollection</span><span class="token punctuation">(</span>“newname”<span class="token punctuation">)</span>

**Q184) what’s the diff between findAndModify and update in MongoDB?**

I’m a little bit confused by the findAndModify method in MongoDB. What’s the advantage of it over update method? For me, it seems that it just returns the item first and then update it.But Why do I need to return the item first? I read the **MongoDB: the definitive guide** and it says that It is handy for manipulating queues and performing other operations that need get-and-set style atomicity. But I didn’t understand how it achieves this. Can somebody explain this to me?  
If you fetch an item and then update it, there may be an update by another thread between those two steps. If you update an item first and then fetch it, there may be another update in-between and you will get back a different item than what you updated.  
Doing it “atomically” means you are guaranteed that you are getting back the exact same item you are updating – i.e. no other operation can happen in between.  
findAndModify returns the document, update does not.  
If I understood Dwight Merriman (one of the original authors of mongoDB) correctly, using update to modify a single document i.e.(“multi”:false} is also atomic. Currently, it should also be faster than doing the equivalent update using findAndModify.  
One useful class of use cases is counters and similar cases. For example, take a look at this code (one of the MongoDB tests): find\_and\_modify4.js.  
Thus, with findAndModify you increment the counter and get its incremented value in one step. Compare: if you (A) perform this operation in two steps and somebody else (B) does the same operation between your steps then A and B may get the same last counter value instead of two different (just one example of possible issues).

**Q185) Getting an error, “Error: couldn’t connect to server 127.0.0.1 shell/mongo.js” & when trying to run mongodb on mac osx lion**

I am using Mac OS X Lion and i just did a fresh install of MongoDB using macports and when i try to run mongodb for the first time i get the following error  
MongoDB shell version: 2.0.5  
connecting to: test  
Fri Jun  1 11:20:33 Error: couldn’t connect to server 127.0.0.1 shell/mongo.js:84  
exception: connect failed  
can anybody please help with this? thanks  
when i run mongod i get:

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19 | Skip code block  hisham-agil:~ hisham$ mongod  mongod –help for help and startup options  Fri Jun  1 11:24:47 [initandlisten] MongoDB starting : pid=53452 port=27017 dbpath=/data/db/ 64-bit host=hisham-agil.local  Fri Jun  1 11:24:47 [initandlisten] db version v2.0.5, pdfile version 4.5  Fri Jun  1 11:24:47 [initandlisten] git version: nogitversion  Fri Jun  1 11:24:47 [initandlisten] build info: Darwin gamma.local 11.3.0 Darwin Kernel Version 11.3.0: Thu Jan 12 18:48:32 PST 2012; root:xnu-1699.24.23~1/RELEASE\_I386 i386 BOOST\_LIB\_VERSION=1\_49  Fri Jun  1 11:24:47 [initandlisten] options: {}  Fri Jun  1 11:24:47 [initandlisten] exception in initAndListen: 10296 dbpath (/data/db/) doesnot exist, terminating  Fri Jun  1 11:24:47 dbexit:  Fri Jun  1 11:24:47 [initandlisten] shutdown: going to close listening sockets…  Fri Jun  1 11:24:47 [initandlisten] shutdown: going to flush diaglog…  Fri Jun  1 11:24:47 [initandlisten] shutdown: going to close sockets…  Fri Jun  1 11:24:47 [initandlisten] shutdown: waiting for fs preallocator…  Fri Jun  1 11:24:47 [initandlisten] shutdown: lock for final commit…  Fri Jun  1 11:24:47 [initandlisten] shutdown: final commit…  Fri Jun  1 11:24:47 [initandlisten] shutdown: closing all files…  Fri Jun  1 11:24:47 [initandlisten] closeAllFiles() finished  Fri Jun  1 11:24:47 dbexit: really exiting now |

You’re running the mongo client without starting the server first. Try running mongod first.  
You’ll have to specify where the mongo “workspace” is using the –dbpath switch, such as mongod –dbpath /mongo/db. The specified folder should exist.  
By default mongod will try to use the directory /data/db for the database files.  
In your case that directory does not exist.  
Before starting mongod you should create those directories and make sure they are writable by the same user that is running mongod process.

**Q186) MongoDB – how to query for a nested item inside a collection?**

I have some data that looks like this:

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16 | Skip code block  [  {  “\_id” : ObjectId(“4e2f2af16f1e7e4c2000000a”),  “advertisers” : [  {  “created\_at” : ISODate(“2011-07-26T21:02:19Z”),  “category” : “Infinity Pro Spin Air Brush”,  “updated\_at” : ISODate(“2011-07-26T21:02:19Z”),  “lowered\_name” : “conair”,  “twitter\_name” : “”,  “facebook\_page\_url” : “”,  “website\_url” : “”,  “user\_ids” : [ ],  “blog\_url” : “”,  }, |

and I was thinking that a query like this would give the id of the advertiser:

|  |  |
| --- | --- |
| 1  2  3 | var start  = new Date(2011, 1, 1);  > var end  = new Date(2011, 12, 12);  > db.agencies.find( { “created\_at” : {$gte : start , $lt : end} } , { \_id : 1 , program\_ids: 1 , advertisers { name : 1 }  } ).limit(1).toArray(); |

But my query didn’t work. Any idea how I can add the fields inside the nested elements to my list of fields I want to get?  
Thanks!  
Use dot notation (e.g. advertisers.name) to query and retrieve fields from nested objects:

|  |  |
| --- | --- |
| 1 | db.agencies.find( { “advertisers.created\_at” : {$gte : start , $lt : end} } , { \_id : 1 , program\_ids: 1 , “advertisers.name”: 1 }  } ).limit(1).toArray(); |

Reference: Retrieving a Subset of Fields and Dot Notation

**Q187) How do I abort a query in mongo javascript shell**

Executing queries in the Javascript shell of Mongo feels a lot like executing commands right in the bash shell. Because of this, my fingers keep trying to Ctrl+C my way out of a command that I want to “abort” and land back at the command prompt. Unfortunately, in the mongo shell at 2.0.4, when I Ctrl+C, it drops me out of the javascript shell, rather than just aborting the command and giving me the command prompt.  
Is there a way in the javascript shell to abandon the current command without executing it and land back at the command prompt?  
What I’ve always naturally done is type a ( and then immediately hit enter three times. The shell will assume you are just hitting enter to break up your command, but if you hit enter 2 more times it will exit back to the shell. It’s wonky, but it works.

**Q188) Where does mongodb stand in the CAP theorem?**

Everywhere I look, I see that MongoDB is CP. But when I dig in I see it is eventually consistent. Is it CP when you use safe=true? If so, does that mean that when I write with safe=true, all replicas will be updated before getting the result?  
MongoDB is strongly consistent by default – if you do a write and then do a read, assuming the write was successful you will always be able to read the result of the write you just read. This is because MongoDB is a single-master system and all reads go to the primary by default. If you optionally enable reading from the secondaries then MongoDB becomes eventually consistent where it’s possible to read out-of-date results.  
MongoDB also gets high-availability through automatic failover in replica sets: https://www.mongodb.org/display/DOCS/Replica+Sets  
This should help answer the question, along with other NoSQL and other persistent storage systems.

**Q189) MongoDB: BIllions of documents in a collection**

I need to load 6.6 billion bigrams into a collection but I can’t find any information on the best way to do this.  
Loading that many documents onto a single primary key index would take forever but as far as I’m aware mongo doesn’t support the equivalent of partitioning?  
Would sharding help? Should I try and split the data set over many collections and build that logic into my application?  
It’s hard to say what the optimal bulk insert is — this party depends on the size of the objects you’re inserting and other immeasurable factors. You could try a few ranges and see what gives you the best performance. As an alternative, some people like using mongoimport, which is pretty fast, but your import data needs to be json or csv. There’s obviously mongodrestore, if the data is in BSON format.  
Mongo can easily handle billions of documents and can have billions of documents in the one collection but remember that the maximum document size is 16mb. There are many folk with billions of documents in MongoDB and there’s lots of discussions about it on the MongoDB Google User Group. Here’s a document on using a large number of collections that you may like to read, if you change your mind and want to have multiple collections instead. The more collections you have, the more indexes you will have also, which probably isn’t what you want.  
Here’s a presentation from Craigslist on inserting billions of documents into MongoDB and the guy’s blogpost.  
It does look like sharding would be a good solution for you but typically sharding is used for scaling across multiple servers and a lot of folk do it because they want to scale their writes or they are unable to keep their working set (data and indexes) in RAM. It is perfectly reasonable to start off with a single server and then move to a shard or replica-set as your data grows or you need extra redundancy and resilience.  
However, there are other users use multiple mongods to get around locking limits of a single mongod with lots of writes. It’s obvious but still worth saying but a multi-mongod setup is more complex to manage than a single server. If your IO or cpu isn’t maxed out here, your working set is smaller than RAM and your data is easy to keep balanced (pretty randomly distributed), you should see improvement (with sharding on a single server). As a FYI, there is potential for memory and IO contention. With 2.2 having improved concurrency with db locking, I suspect that there will be much less of a reason for such a deployment.  
You need to plan your move to sharding properly, i.e. think carefully about choosing your shard key. If you go this way then it’s best to pre-split and turn off the balancer. It will be counter-productive to be moving data around to keep things balanced which means you will need to decide up front how to split it. Additionally, it is sometimes important to design your documents with the idea that some field will be useful for sharding on, or as a primary key.  
Here’s some good links –  
1. Choosing a Shard Key  
2. Blog post on shard keys  
3. Overview presentation on sharding  
4. Presentation on Sharding Best Practices  
You can absolutely shard data in MongoDB (which partitions across N servers on the shard key). In fact, that’s one of it’s core strengths. There is no need to do that in your application.  
For most use cases, I would strongly recommend doing that for 6.6 billion documents. In my experience, MongoDB performs better with a number of mid-range servers rather than one large one.

**Q190) Why are key names stored in the document in MongodDB**

I’m curious about this quote from Kyle Banker’s MongoDB In Action:  
Its important to consider the length of the key names you choose, since key names are stored in the documents themselves. This contrasts with an RDBMS, where column names are always kept separate from the rows they refer to. So when using BSON, if you can live with dob in place of date\_of\_birth as a key name, youll save 10 bytes per document. That may not sound like much, but once you have a billion such documents, youll have saved nearly 10 GB of storage space just by using a shorter key name. This doesnt mean you should go to unreasonable lengths to ensure small key names; be sensible. But if you expect massive amounts of data, economizing on key names will save space.  
I am interested in the reason why this is not optimized on the database server side. Would a in-memory lookup table with all key names in the collection be too much of a performance penalty that is not worth the potential space savings?  
What you are referring to is often called “key compression”\*. There are several reasons why it hasn’t been implemented:  
1. If you want it done, you can currently do it at the Application/ORM/ODM level quite easily.  
2. It’s not necessarily a performance\*\* advantage in all cases think collections with lots of key names, and/or key names that vary wildly between documents.  
3. It might not provide a measurable performance\*\* advantage at all until you have millions of documents.  
4. If the server does it, the full key names still have to be transmitted over the network.  
5. If compressed key names are transmitted over the network, then readabilityreally suffers using the javascript console.  
6. Compressing the entire JSON document might offer an even better performance advantage.  
Like all features, there’s a cost benefit analysis for implementing it, and (at least so far) other features have offered more “bang for the buck”.  
Full document compression is being considered for a future MongoDB version.  
\* An in-memory lookup table for key names is basically a special case of LZW style compression that’s more or less what most compression algorithms do.  
\*\* Compression provides both a space advantage and a performance advantage. Smaller documents means that more documents can be read per IO, which means that in a system with fixed IO, more documents per second can be read.  
I believe one of the original reasons behind storing the key names with the documents is to allow a more easily scalable schema-less database. Each document is self-contained to a greater extent, in that if you move the document to another server (for example, via replication or sharding) you can index the contents of the document without having to reference separate or centralized metadata such as a mapping of key names to more compact key IDs.  
Since there is no enforced schema for a MongoDB collection, the field names can potentially be different for every document in the same collection. In a sharded environment, inserts to each shard are (intentionally) independent so at a document level the raw data could end up differing unless the key mapping was able to be consistent per shard.  
Depending on your use case, the key names may or may not consume a significant amount of space relative to the accompanying data. You could always workaround the storage concern from the application / ODM implementation by mapping YourFriendlyKeyNames to shorter DB key equivalents.  
There is an open MongoDB Jira issue and some further discussion to have the server tokenize field names, which you can vote on to help prioritize including this feature in a future release.  
MongoDB’s current design goals include performance with dynamic schemas, replication & high availability, auto-sharding, and in-place updates .. with one potential tradeoff being some extra disk usage.

**Q191) How can I copy collection to another database in MongoDB?**

Is there some simple way to fulfill this?  
At the moment there is no command in MongoDB that would do this. Please note the JIRA ticket with related feature request: https://jira.mongodb.org/browse/SERVER-732.  
You could do something like:  
db..find().forEach(function(d){ db.getSiblingDB(‘’)[‘’].insert(d); });  
Please note that with this, the two databases would need to share the same mongod for this to work.  
Besides this, you can do a mongodump of a collection from one database and then mongorestore the collection to the other database.  
The best way is to do a mongodump then mongorestore.  
You can select the collection via:  
mongodump -d some\_database -c some\_collection  
[Optionally, zip the dump (zip some\_database.zip some\_database/\* -r) and scp it elsewhere]  
Then restore it:  
mongorestore -d some\_other\_db -c some\_or\_other\_collection dump/some\_collection.bson  
Existing data in some\_or\_other\_collection will be preserved. That way you can “append” a collection from one database to another.  
Prior to version 2.4.3, you will also need to add back your indexes after you copy over your data. Starting with 2.4.3, this process is automatic, and you can disable it with –noIndexRestore.  
I would abuse the connect function in mongo cli mongo doc. so that means you can start one or more connection. if you want to copy customer collection from test to test2 in same server. first you start mongo shell  
use test  
var db2 = connect(‘localhost:27017/test2’)  
do a normal find and copy the first 20 record to test2.  
db.customer.find().limit(20).forEach(function(p) { db2.customer.insert(p); });  
or filter by some criteria  
db.customer.find({“active”: 1}).forEach(function(p) { db2.customer.insert(p); });  
just change the localhost to IP or hostname to connect to remote server. I use this to copy test data to a test database for testing.  
Actually, there is a command to move a collection from one database to another. It’s just not called “move” or “copy”.  
To copy a collection, you can clone it on the same db, then move the clone.  
To clone:

|  |  |  |
| --- | --- | --- |
| 1  2  3  4  5  6  7  8 | > use db1  > db.source\_collection.find().forEach( function(x){db.collection\_copy.insert(x)} );  To move:  > use admin  switched to db admin  > db.runCommand({renameCollection: ‘db1.source\_collection, to: ‘db2.target\_collection’}) // who’d think rename could move?  The other answers are better for copying the collection, but this is especially useful if you’re looking to move it.  I’d usually do: | |
| 1  2  3  4 | use sourcedatabase;  var docs=db.sourcetable.find();  use targetdatabase;  docs.forEach(function(doc) { db.targettable.insert(doc); }); |

**Q192) In Mongo what is the difference between sharding and replication?**

Replication seems to be a lot simpler than sharding, unless I am missing the benefits of what sharding is actually trying to achieve. Don’t they both provide horizontal scaling?  
In the context of scaling MongoDB:  
replicationcreates additional copies of the data and allows for automatic failover to another node. 1. Replication may help with horizontal scaling of reads if you are OK to read data that potentially isn’t the latest.  
2. sharding allows for horizontal scaling of data writes by partitioning data across multiple servers using a shard key. It’s important to choose a good shard key. For example, a poor choice of shard key could lead to “hot spots” of data only being written on a single shard.  
A sharded environment does add more complexity because MongoDB now has to manage distributing data and requests between shards — additional configuration and routing processes are added to manage those aspects.  
Replication and sharding are typically combined to created a sharded cluster where each shard is supported by a replica set.  
From a client application point of view you also have some control in relation to the replication/sharding interaction, in particular:  
1. Read preferences  
2. Write concerns  
Replication is a mostly traditional master/slave setup, data is synced to backup members and if the primary fails one of them can take its place. It is a reasonably simple tool. It’s primarily meant for redundancy, although you can scale reads by adding replica set members. That’s a little complicated, but works very well for some apps.  
Sharding sits on top of replication, usually. “Shards” in MongoDB are just replica sets with something called a “router” in front of them. Your application will connect to the router, issue queries, and it will decide which replica set (shard) to forward things on to. It’s significantly more complex than a single replica set because you have the router and config servers to deal with (these keep track of what data is stored where).  
If you want to scale Mongo horizontally, you’d shard. 10gen likes to call the router/config server setup auto-sharding. It’s possible to do a more ghetto form of sharding where you have the app decide which DB to write to as well.

**Q193) What does Mongo sort on when no sort order is specified?**

When we run a Mongo find() query without any sort order specified, what does the database internally use to sort the results?  
According to the documentation on the mongo website:  
When executing a find() with no parameters, the database returns objects in forward natural order.  
For standard tables, natural order is not particularly useful because, although the order is often close to insertion order, it is not guaranteed to be. However, for Capped Collections, natural order is guaranteed to be the insertion order. This can be very useful.  
However for standard collections (non capped collections), what field is used to sort the results? Is it the **\_id**field or something else?  
**Edit:**  
Basically, I guess what I am trying to get at is that if I execute the following search query:  
db.collection.find({“x”:y}).skip(10000).limit(1000);  
At two different points in time: **t1** and**t2**, will I get different result sets:  
1. When there have been no additional writes between t1 & t2?  
2. When there have been new writes between t1 & t2?  
3. There are new indexes that have been added between t1 & t2?  
I have run some tests on a temp database and the results I have gotten are the same (**Yes**) for all the 3 cases – but I wanted to be sure and I am certain that my test cases weren’t very thorough.  
By definition the sort **defaults to undefined**, and so is the return order of documents. If there is no query then it will use the natural order. The results are returned in the **order they are found,** which may coincide with insertion order (but isn’t guaranteed to be) or the order of the index(es) used.  
Some examples that will affect storage (natural) order:  
1. if documents are updated and don’t fit in their currently allocated space, they will be moved  
new documents may be inserted in available gaps created by deleted or moved documents  
2. If an index is used, docs will be returned in the order they are found. If more than one index is used then the order depends internally on which index first identified the document during the de-duplication process.  
If you**want a specific order**then you **must**include a sort with your query.  
The exception noted for capped collections’ natural order is because documents can’t move and are stored in insertion order. The ordering is part of the capped collection feature that ensures the oldest documents “age out” first. Additionally, documents cannot be deleted or moved in a capped collection (see Usage and Restrictions for more info).

**Q194) How to reclaiming deleted space without `db.repairDatabase()`?**

I want to shrink data files size by reclaiming deleted space, but I can’t run db.repairDatabase(), because free disk space is not enough.  
The original answer to this question is here: Reducing MongoDB database file size  
There really is nothing outside of repair that will reclaim space. The compact should allow you to go much longer on the existing space. Otherwise, you will have to migrate to a bigger drive.  
One way to do this is to use an off-line secondary from your Replica Set. This should give you a whole maintenance window to migrate, repair, move back and bring back up.  
If you are not running a Replica Set, it’s time to look at doing just that.  
You could run the compact command on a single collection, or one by one in all the collections you want to shrink.  
https://www.mongodb.org/display/DOCS/Compact+Command  
db.runCommand( { compact : ‘mycollectionname’ } )  
As noted in comments, I was mistaken, compact does not actually reclaim disk space, it only defragments and rebuilds collection indexes.  
Instead though, you could use “–repairpath” option if you have another drive available which has available freespace.  
For example:  
mongod –dbpath /data/db –repair –repairpath /data/db0  
Shown here: https://docs.mongodb.org/manual/tutorial/recover-data-following-unexpected-shutdown/

**Q195) creating a different database for each collection in MongoDB 2.2**

MongoDB 2.2 has a write lock per database as opposed to a global write lock on the server in previous versions. So would it be ok if i store each collection in a separate database to effectively have a write lock per collection.(This will make it look like MyISAM’s table level locking). Is this approach faulty?  
There’s a key limitation to the locking and that is the local database. That database includes a theoplog collection which is used for replication.  
If you’re running in production, you should be running with Replica Sets. If you’re running with Replica Sets, you need to be aware of the write lock effect on that database.  
Breaking out your 10 collections into 10 DBs is useless if they all block waiting for the oplog.  
Before taking a large step to re-write, please ensure that the oplog will not cause issues.  
Also, be aware that MongoDB implements DB-level security. If you’re using any security features, you are now creating more DBs to secure.

**Q196) How to stop mongo DB in one command**

I need to be able to start/stop MongoDB on the cli. It is quite simple to start:  
./mongod  
But to stop mongo DB, I need to run open mongo shell first and then type two commands:  
$ ./mongo  
use admin  
db.shutdownServer()  
So I don’t know how to stop mongo DB in one line. Any help?  
Starting and Stopping MongoDB is covered in the MongoDB manual. This section of the manual explains the various options of stopping MongoDB through the shell, cli, drivers etc. It also details the risks of incorrectly stopping MongoDB (such as data corruption) and talks about the different kill signals.  
Additionally, if you have installed MongoDB using a package manager for Ubuntu or Debian then you can stop mongod as follows:

1. Upstart:sudo service mongodb stop<br>2. Sysvinit:sudo /etc/init.d/mongodb stop

Or on Red Hat based systems:  
1. service mongod stop  
Or on Windows if you have installed as a service named MongoDB:  
1. net stop MongoDB  
To learn more about the problems of an unclean shutdown, how to best avoid such a scenario and what to do in the event of an unclean shutdown, please see: Recover Data after an Unexpected Shutdown.  
If you literally want a one line equivalent to the commands in your original question, you could alias:  
mongo –eval “db.getSiblingDB(‘admin’).shutdownServer()”  
Mark’s answer on starting and stopping MongoDB via services is the more typical (and recommended) administrative approach.

|  |  |
| --- | --- |
| 1 | mongod –dbpath /path/to/your/db –shutdown |

More info at official: https://docs.mongodb.org/manual/tutorial/manage-mongodb-processes/

**Q197) MongoDB select count(distinct x) on an indexed column – count unique results for large data sets**

I have gone through several articles and examples, and have yet to find an efficient way to do this SQL query in MongoDB (where there are millions of  documents)  
**First attempt**

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8 | (e.g. from this almost duplicate question – Mongo equivalent of SQL’s SELECT DISTINCT?)  db.myCollection.distinct(“myIndexedNonUniqueField”).length  Obviously I got this error as my dataset is huge  Thu Aug 02 12:55:24 uncaught exception: distinct failed: {  “errmsg” : “exception: distinct too big, 16mb cap”,  “code” : 10044,  “ok” : 0  } |

**Second attempt**  
I decided to try and do a group  
db.myCollection.group({key: {myIndexedNonUniqueField: 1},  
initial: {count: 0},  
reduce: function (obj, prev) { prev.count++;} } );  
But I got this error message instead:  
exception: group() can’t handle more than 20000 unique keys  
**Third attempt**

<br>I haven’t tried yet but there are several suggestions that involve mapReduce<br>e.g.<br>1. this onehow to do distinct and group in mongodb? (not accepted, answer author / OP didn’t test it)<br>2. this oneMongoDB group by Functionalities (seems similar to Second Attempt)<br>3. this onehttps://blog.emmettshear.com/post/2010/02/12/Counting-Uniques-With-MongoDB<br>4. this one https://groups.google.com/forum/?fromgroups#!topic/mongodb-user/trDn3jJjqtE<br>5. this onehttps://cookbook.mongodb.org/patterns/unique\_items\_map\_reduce/

**Also**  
It seems there is a pull request on GitHub fixing the .distinct method to mention it should only return a count, but it’s still open: https://github.com/mongodb/mongo/pull/34  
But at this point I thought it’s worth to ask here, what is the latest on the subject? Should I move to SQL or another NoSQL DB for distinct counts? or is there an efficient way?  
**Update:**  
This comment on the MongoDB official docs is not encouraging, is this accurate?

https://www.mongodb.org/display/DOCS/Aggregation#comment-430445808

**Update2:**  
Seems the new Aggregation Framework answers the above comment… (MongoDB 2.1/2.2 and above, development preview available, not for production)  
https://docs.mongodb.org/manual/applications/aggregation/

1) The easiest way to do this is via the aggregation framework. This takes two “$group” commands: the first one groups by distinct values, the second one counts all of the distinct values

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15 | Skip code block  pipeline = [  { $group: { \_id: “$myIndexedNonUniqueField”}  },  { $group: { \_id: 1, count: { $sum: 1 } } }  ];  //  // Run the aggregation command  //  R = db.runCommand(  {  “aggregate”: “myCollection” ,  “pipeline”: pipeline  }  );  printjson(R); |

2) If you want to do this with Map/Reduce you can. This is also a two-phase process: in the first phase we build a new collection with a list of every distinct value for the key. In the second we do a count() on the new collection.

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24 | Skip code block  var SOURCE = db.myCollection;  var DEST = db.distinct  DEST.drop();  map = function() {  emit( this.myIndexedNonUniqueField , {count: 1});  }  reduce = function(key, values) {  var count = 0;  values.forEach(function(v) {  count += v[‘count’];        // count each distinct value for lagniappe  });  return {count: count};  };  //  // run map/reduce  //  res = SOURCE.mapReduce( map, reduce,  { out: ‘distinct’,  verbose: true  }  );  print( “distinct count= ” + res.counts.output );  print( “distinct count=”, DEST.count() ); |

Note that you cannot return the result of the map/reduce inline, because that will potentially overrun the 16MB document size limit. You can save the calculation in a collection and then count() the size of the collection, or you can get the number of results from the return value of mapReduce().

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8 | db.myCollection.aggregate(  {$group : {\_id : “$myIndexedNonUniqueField”} },  {$group: {\_id:1, count: {$sum : 1 }}});  straight to result:  db.myCollection.aggregate(  {$group : {\_id : “$myIndexedNonUniqueField”} },  {$group: {\_id:1, count: {$sum : 1 }}})  .result[0].count; |

**Q198) MongoDB: Find Subdocument in Array Matching Parameters**

In MongoDB I would like to find a document based on the values of a subdocument meeting certain parameters. Specifically I have a document structured like this:

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14 | Skip code block  {  name: “test”,  data: [{  name: “test1”,  start: 0,  end: 2  },  {  name: “test2”,  start: 15  end: 18  }]  } |

How can I tell MongoDB to only return my document if the start time for a data subdocument is less than 5 and the end time for the same subdocument is greater than 5? Currently, if I do

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11 | db.foo.findOne({  ‘data.start’: { $lte: 5 },  ‘data.end’: { $gte: 5 }  })  it will return my document always because 5 is greater than 0 and less than 18. How can I tell MongoDB to only return my document if 5 (or whatever value) is greater than 0 and less than 2 OR greater than 15 and less than 18?  You want to use $elemMatch.  db.foo.findOne({ data: { $elemMatch : {  start: { $lte: 5 },  end: { $gte: 5 }  }}  }) |

**Q199) How to efficiently perform “distinct” with multiple keys?**

For example, there is a collection like this:

|  |  |
| --- | --- |
| 1  2  3  4  5 | {market: ‘SH’, code: ‘000001’, date: ‘2012-01-01’, price: 1000}  {market: ‘SZ’, code: ‘000001’, date: ‘2012-01-01’, price: 1000}  {market: ‘SH’, code: ‘000001’, date: ‘2012-01-02’, price: 1000}  {market: ‘SZ’, code: ‘000001’, date: ‘2012-01-02’, price: 1000}  {market: ‘SH’, code: ‘000002’, date: ‘2012-01-03’,price: 1000} |

…  
This collection contains tens of millions documents.  
I want to call distinct with two keys:

|  |  |
| --- | --- |
| 1 | collection.distinct(‘market’, ‘code’); |

and get result :

|  |  |
| --- | --- |
| 1 | [{market: ‘SH’, code:’000001'}, {market: ‘SZ’, code:’000001'}, {market: ‘SH’, code:’000002'}] |

As native distinct command accept only one key, I try to implement it by using map-reduce. But map-reduce is far too slow to native distinct. In my one-key distinct test, map-reduce spend about ten times longer than native distinct.  
Is there a efficient way to implement multikey distinct?  
If you are willing to wait for the upcoming 2.2 release of MongoDB, you can run this query efficiently using the aggregation framework:

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7 | collection = db.tb;  result = collection.aggregate(  [  {“$group”: { “\_id”: { market: “$market”, code: “$code” } } }  ]  );  printjson(result); |

On a million-record collection on my test machine, this ran in 4 seconds, while the map/reduce version took over a minute.

**Q200) MongoDB: Trade-offs of dropping a collection vs. removing all of its documents**

What are the trade-offs of dropping a MongoDB collection vs. removing all of its documents (assuming the collection will be re-created immediately)?  
If you go through a remove all the documents from a collection, then you’ll be doing a lot more work (freeing the document’s storage, clearing the index entries that point to the document, and so on). If you instead just drop the collection, it’ll just be reclaiming the extents that the collection and its indexes use.  
One other difference is that dropping the collection will also remove the collection’s indexes.  
A benefit of simply dropping a collection is that it is much faster than removing all of a collection’s documents. If your collection will be “re-created immediately” anyway (assuming that includes index re-creation), then this is probably the most-attractive option.  
The authors of the book MongoDB: The Definitive Guide (Kristina Chodorow and Michael Dirolf) ran an experiment where they provided a Python script which timed a drop vs. a remove of 1000000 records. The results came in at 46.08 seconds for the remove and .01 seconds for the drop. Now while the exact times may differ based-on hardware and other factors, it nonetheless illustrates the point that the drop is significantly faster.  
reference: Chodorow K., Dirolf M. (2010). MongoDB: The Definitive Guide. O’Reilly Media, Inc. Sebastapol, CA., pp.25

**Related Page:**[**MongoDB Vs MySQL - Which Is A Better Database?**](https://mindmajix.com/mongodb-vs-mysql)

**Q201) MongoDB : find value in Array with multiple criteria**

I have the following documents:

|  |  |
| --- | --- |
| 1  2 | {\_id : 1, numbers : [-1000, 1000]}  {\_id : 2, numbers : [5]} |

I’m trying to get a query that will find a document that has a value in the numbers array that is between -10 and 10 (in this case, \_id : 2). However, when I try the following:

|  |  |
| --- | --- |
| 1 | db.foo.find({numbers : $and : [{$gt : -10},{$lt : 10}]}) |

it returns all documents. Is this possible to do without map-reduce? Thanks, -JWW  
You can use $elemMatch to check if an element in an array matches a specified match expression.  
In this case, you can use it to get a document whose numbers array has an element that is between -10 and 10:

|  |  |
| --- | --- |
| 1 | db.foo.find( { numbers : { $elemMatch : { $gt : -10 , $lt : 10 } } } ); |

This will just return the \_id : 2 document.

**Q202) MongoDB 2.1 Aggregate Framework Sum of Array Elements matching a name**

This is a question about the best way to add up a series of data in an array where I have to match another element. I’m trying to use the 2.2 Aggregation framework and it’s possible I can do this with a simple group.  
So for a given set of documents I’m trying to get an output like this;

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25  26  27  28  29  30  31 | Skip code block  {  “result” : [  {  “\_id” : null,  “numberOf”: 2,  “Sales” : 468000,  “profit” : 246246,  }  ],  “ok” : 1  }  Now, I originally had a list of documents, containing values assigned to named properties, eg;  Skip code block  [  {  \_id : 1,  finance: {  sales: 234000,  profit: 123123,  }  }  ,  {  \_id : 2,  finance: {  sales: 234000,  profit: 123123,  }  }  ] |

This was easy enough to add up, but the structure didn’t work for other reasons. For instance, there are may other columns like “finance” and I want to be able to index them without creating thousands of indexes, so I need to convert to a structure like this;

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25  26  27  28  29  30  31  32  33  34  35  36  37  38  39  40  41  42 | Skip code block  [  {  \_id : 1,  finance: [  {  “k”: “sales”,  “v”: {  “description”:”sales over the year”,  v: 234000,  }  },  {  “k”: “profit”,  “v”: {  “description”:”money made from sales”,  v: 123123,  }  }  ]  }  ,  {  \_id : 2,  finance: [  {  “k”: “sales”,  “v”: {  “description”:”sales over the year”,  v: 234000,  }  },  {  “k”: “profit”,  “v”: {  “description”: “money made from sales”,  v: 123123,  }  }  ]  }  ] |

I can index finance.k if I want, but then I’m struggling to build an aggregate query to add up all the numbers matching a particular key. This was the reason I originally went for named properties, but this really needs to work in a situation whereby there are thousands of “k” labels.  
Does anyone know how to build an aggregate query for this using the new framework? I’ve tried this;

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25  26  27  28  29  30  31  32  33  34  35  36  37  38  39  40  41  42  43  44  45  46  47  48  49  50  51  52  53  54  55  56  57  58  59  60  61  62  63 | Skip code block  db.projects.aggregate([  {  $match: {  // QUERY  $and: [  // main query  {},  ]  }  },  {  $group: {  \_id: null,  “numberOf”: { $sum: 1 },  “sales”:    { $sum: “$finance.v.v” },  “profit”:   { $sum: “$finance.v.v” },  }  },  ])  but I get;  {  “errmsg” : “exception: can’t convert from BSON type Array to double”,  “code” : 16005,  “ok” : 0  }  \*\* For extra kudos, I’ll need to be able to do this in a MapReduce query as well.  You can use the aggregation framework to get sales and profit and any other value you may be storing in your key/value pair representation.  For your example data:  Skip code block  var pipeline = [  {  “$unwind” : “$finance”  },  {  “$group” : {  “\_id” : “$finance.k”,  “numberOf” : {  “$sum” : 1  },  “total” : {  “$sum” : “$finance.v.v”  }  }  }  ]  R = db.tb.aggregate( pipeline );  printjson(R);  {  “result” : [  {  “\_id” : “profit”,  “numberOf” : 2,  “total” : 246246  },  {  “\_id” : “sales”,  “numberOf” : 2,  “total” : 468000  }  ],  “ok” : 1  } |

If you have additional k/v pairs then you can add a match which only passes through k values in [“sales”,”profit”].  
You will have to use ‘$unwind” to break out the values in the array, which will mean that you can’t get the sum of the sales and the profit in a single aggregation command. Given that, the query itself is easy:

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25  26  27  28  29  30  31  32  33  34  35  36  37  38  39  40  41  42  43  44  45  46  47  48 | Skip code block  var pipeline = [  {“$unwind”: “$finance” } ,  {“$match”: {“finance.k”: “sales”} },  { $group:  { \_id: null,  numberOf: { “$sum”: 1 },  sales: {“$sum”: “$finance.v.v” }  }  }  ];  R = db.tb.aggregate( pipeline );  printjson(R);  {  “result” : [  {  “\_id” : null,  “numberOf” : 2,  “sales” : 236340  }  ],  “ok” : 1  }  You can run a similar query for profit, just substitute “profit” for “sales” in the “$match” operator.  Oh, and here’s the map/reduce example:  Skip code block  map = function() {  var ret = { sales: 0.0 , profit: 0.0, count: 1 };  // iterate over ‘finance[]’ array  this.finance.forEach( function (i) {  if ( i.k == “sales” ) ret.sales =  i.v.v ;  if ( i.k == “profit” ) ret.profit =  i.v.v ;  } );  emit( 1, ret );  }  reduce = function(key, values) {  var ret = { sales: 0.0 , profit: 0.0, count: 0 };  values.forEach(function(v) {  ret.sales += v.sales;  ret.profit += v.profit;  ret.count += v.count;  });  return ret;  };  //  // run map/reduce  //  res = SOURCE.mapReduce( map, reduce ); |

**Q203) Select \* group by in mongo aggregation**

I am trying to do something that I think is quite simple. Suppose I have a series of records in mongo that have a common key, and variable number of attributes. I want to select all attributes and group by name across the records. For example  
  
Tried

|  |  |
| --- | --- |
| 1 | DB.data.aggregate({ $group : { \_id : “$Name” } }) |

Unfortunately I get back all the names as records but not the union of all the possible attributes.  
If you want to combine the attributes, you’ll need to add those to the group. For example, using$addToSet to find the unique values of the x,y,z attributes grouped by each name:

|  |  |
| --- | --- |
| 1  2 | db.data.aggregate(  { $group : { |
| 1  2  3  4  5  6  7 | { Name: George, x: 5, y: 3 }  { Name: George, z: 9 }  { Name: Rob, x: 12, y: 2 }  I would like to produce a CSV that looks like this:  Name     X   Y   Z  George   5   3   9  Rob      12  2 |

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25  26  27  28  29  30  31  32  33  34  35 | \_id : “$Name”,  x: { $addToSet: “$x” },  y: { $addToSet: “$y” },  z: { $addToSet: “$z” },  }}  )  Returns:  Skip code block  {  “result” : [  {  “\_id” : “Rob”,  “x” : [  12  ],  “y” : [  2  ],  “z” : [ ]  },  {  “\_id” : “George”,  “x” : [  5  ],  “y” : [  3  ],  “z” : [  9  ]  }  ],  “ok” : 1  } |

**Q204) Creating custom Object ID in MongoDB**

I am creating a service for which I will use MongoDB as a storage backend. The service will produce a hash of the user input and then see if that same hash (+ input) already exists in our dataset.

The hash will be unique yet random ( = non-incremental/sequential), so my question is:<br>Is it -legitimate- to use a random value for an Object ID? Example:<br>$object\_id = new MongoId(HEX-OF-96BIT-HASH);

Or will MongoDB treat the ObjectID differently from other server-produced ones, since a “real” ObjectID also contains timestamps, machine\_id, etc?  
What are the pros and cons of using a ‘random’ value? I guess it would be statistically slower for the engine to update the index on inserts when the new \_id’s are not in any way incremental – am I correct on that?  
Yes it is perfectly fine to use a random value for an object id, if some value is present in \_id field of a document being stored, it is treated as objectId.  
Since \_id field is always indexed, and primary key, you need to make sure that different objectid is generated for each object. There are some guidelines to optimize user defined object ids :

https://www.mongodb.org/display/DOCS/Optimizing+Object+IDs#OptimizingObjectIDs-Usethecollections%27naturalprimarykey%27intheidfield.

Whether it is good or bad depends upon it’s uniqueness. Of course the ObjectId provided by MongoDB is quite unique so this is a good thing. So long as you can replicate that uniqueness then you should be fine.  
There are no inherent risks/performance loses by using your own ID. I guess using it in string form might use up more index/storage/querying power but there you are using it in MongoID (ObjectId) form which should preserve the strengths of not storing it in a simple string.  
While any values, including hashes, can be used for the \_id field, I would recommend against using random values for two reasons:  
1. You may need to develop a collision-management strategy in the case you produce identical random values for two different objects. In the question, you imply that you’ll generate IDs using a some type of a hash algorithm. I would not consider these values “random” as they are based on the content you are digesting with the hash. The probability of a collision then is a function of the diversity of content and the hash algorithm. If you are using something like MD5 or SHA-1, I wouldn’t worry about the algorithm, just the content you are hashing. If you need to develop a collision-management strategy then you definitely should not use random or hash-based IDs as collision management in a clustered environment is complicated and requires additional queries.  
2. Random values as well as hash values are purposefully meant to be dispersed on the number line. That (a) will require more of the B-tree index to be kept in memory at all times and (b) may cause variable insert performance due to B-tree rebalancing. MongoDB is optimized to handle ObjectIDs, which come in ascending order (with one second time granularity). You’re likely better off sticking with them.

**Q205) How to resolve error :dbpath (/data/db/) does not exist permanently in MongoDB**

I have installed mongodb in my Ubuntu 10.04.  
I know that when it comes to start the mongodb server with the command “mongod“,then it expects /data/db folder and it can be easily resolved by creating “/data/db/”. One more way is to provide your own path using mongod –dbpath “path”,when we intend to give our own custom path for db.  
But while going through https://docs.mongodb.org/manual/tutorial/install-mongodb-on-ubuntu/link i found that there is a configuration file.  
I made the following changes to it.

|  |  |
| --- | --- |
| 1  2  3  4 | # mongodb.conf  dbpath=/EBS/Work/mongodb/data/db/  logpath=/EBS/Work/mongodb/mongodb.log  logappend=true |

But still when I try to start the server with “**mongod**” it throws the same **error i.e error :dbpath (/data/db/) does not exist** . I wanted to know that how can I permanently redirect my dbpath to my own custom folder cause everytime you don’t want to type the path using “mongod –dbpath path”.Rather we look to make some changes in configuration file.  
Assuming you have followed the instructions to install a packaged version of MongoDB from 10gen, you should be starting and stopping mongod using service.  
To start mongod:  
sudo service mongodb start  
To stop mongod:  
sudo service mongodb stop  
If you use the service command to start and stop, it should be using the configuration file:/etc/mongodb.conf.  
**Starting mongod from the command line**  
If you run mongod directly instead of using the service definition, you will also have to specify a configuration file as a command line parameter if you want one to be used:  
mongod –config /etc/mongodb.conf  
here is how i solve this problem. followed their offcial Doc . HERE. long story short , I created a directory inside /srv and excuted the command mongod –dbpath /srv/mongodb/ RAN like a champ.

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25  26  27  28  29  30  31  32  33  34  35  36  37  38  39  40  41  42  43  44  45  46  47  48  49  50  51  52  53  54  55  56  57  58  59  60  61  62  63  64  65  66  67  68  69  70  71  72  73  74  75 | Skip code block  ~$ mongod  Tue Jun  3 20:27:39.564 [initandlisten] MongoDB starting : pid=5380 port=27017 dbpath=/srv/mongodb/ 64-bit host= -SVE1411EGXB  Tue Jun  3 20:27:39.564 [initandlisten] db version v2.4.10  Tue Jun  3 20:27:39.564 [initandlisten] git version: e3d78955d181e475345ebd60053a4738a4c5268a  Tue Jun  3 20:27:39.564 [initandlisten] build info: Linux ip-10-2-29-40 2.6.21.7-2.ec2.v1.2.fc8xen #1 SMP Fri Nov 20 17:48:28 EST 2009 x86\_64 BOOST\_LIB\_VERSION=1\_49  Tue Jun  3 20:27:39.564 [initandlisten] allocator: tcmalloc  Tue Jun  3 20:27:39.564 [initandlisten] options: { dbpath: “/srv/mongodb/” }  Tue Jun  3 20:27:39.565 [initandlisten] exception in initAndListen: 10296  \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  ERROR: dbpath (/srv/mongodb/) does not exist.  Create this directory or give existing directory in –dbpath.  See https://dochub.mongodb.org/core/startingandstoppingmongo  \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  , terminating  Tue Jun  3 20:27:39.565 dbexit:  Tue Jun  3 20:27:39.565 [initandlisten] shutdown: going to close listening sockets…  Tue Jun  3 20:27:39.565 [initandlisten] shutdown: going to flush diaglog…  Tue Jun  3 20:27:39.565 [initandlisten] shutdown: going to close sockets…  Tue Jun  3 20:27:39.565 [initandlisten] shutdown: waiting for fs preallocator…  Tue Jun  3 20:27:39.565 [initandlisten] shutdown: lock for final commit…  Tue Jun  3 20:27:39.565 [initandlisten] shutdown: final commit…  Tue Jun  3 20:27:39.565 [initandlisten] shutdown: closing all files…  Tue Jun  3 20:27:39.565 [initandlisten] closeAllFiles() finished  Tue Jun  3 20:27:39.565 dbexit: really exiting now  ~$ mongod –dbpath /srv/mongodb/  Tue Jun  3 20:27:55.616 [initandlisten] MongoDB starting : pid=5445 port=27017 dbpath=/srv/mongodb/ 64-bit host= -SVE1411EGXB  Tue Jun  3 20:27:55.616 [initandlisten] db version v2.4.10  Tue Jun  3 20:27:55.616 [initandlisten] git version: e3d78955d181e475345ebd60053a4738a4c5268a  Tue Jun  3 20:27:55.616 [initandlisten] build info: Linux ip-10-2-29-40 2.6.21.7-2.ec2.v1.2.fc8xen #1 SMP Fri Nov 20 17:48:28 EST 2009 x86\_64 BOOST\_LIB\_VERSION=1\_49  Tue Jun  3 20:27:55.616 [initandlisten] allocator: tcmalloc  Tue Jun  3 20:27:55.616 [initandlisten] options: { dbpath: “/srv/mongodb/” }  Tue Jun  3 20:27:55.617 [initandlisten] exception in initAndListen: 10296  ~$ sudo service mongodb start  start: Job is already running: mongodb  ~$ sudo service mongodb stop  mongodb stop/waiting  ~$ cd /srv/  ~$~$/srv$ ls  ~$ /srv$ mkdir mongodb  mkdir: cannot create directory mongodb: Permission denied  ~$ cd –  /srv  ~$ sudo chgrp   /srv  ~$ sudo chmod 775 /srv  ~$ cd /srv/  /srv$ ls  /srv$ mkdir mongodb  /srv$ ls mongodb  /srv$ cd  ~$ mongod –dbpath /srv/mongodb/  Tue Jun  3 20:40:57.457 [initandlisten] MongoDB starting : pid=6018 port=27017 dbpath=/srv/mongodb/ 64-bit host= -SVE1411EGXB  Tue Jun  3 20:40:57.457 [initandlisten] db version v2.4.10  Tue Jun  3 20:40:57.457 [initandlisten] git version: e3d78955d181e475345ebd60053a4738a4c5268a  Tue Jun  3 20:40:57.457 [initandlisten] build info: Linux ip-10-2-29-40 2.6.21.7-2.ec2.v1.2.fc8xen #1 SMP Fri Nov 20 17:48:28 EST 2009 x86\_64 BOOST\_LIB\_VERSION=1\_49  Tue Jun  3 20:40:57.457 [initandlisten] allocator: tcmalloc  Tue Jun  3 20:40:57.457 [initandlisten] options: { dbpath: “/srv/mongodb/” }  Tue Jun  3 20:40:57.520 [initandlisten] journal dir=/srv/mongodb/journal  Tue Jun  3 20:40:57.521 [initandlisten] recover : no journal files present, no recovery needed  Tue Jun  3 20:41:00.545 [initandlisten] preallocateIsFaster=true 36.86  Tue Jun  3 20:41:03.489 [initandlisten] preallocateIsFaster=true 35.06  Tue Jun  3 20:41:07.456 [initandlisten] preallocateIsFaster=true 34.44  Tue Jun  3 20:41:07.456 [initandlisten] preallocateIsFaster check took 9.935 secs  Tue Jun  3 20:41:07.456 [initandlisten] preallocating a journal file /srv/mongodb/journal/prealloc.0  Tue Jun  3 20:41:10.009 [initandlisten]         File Preallocator Progress: 985661440/1073741824    91%  Tue Jun  3 20:41:22.273 [initandlisten] preallocating a journal file /srv/mongodb/journal/prealloc.1  Tue Jun  3 20:41:25.009 [initandlisten]         File Preallocator Progress: 933232640/1073741824    86%  Tue Jun  3 20:41:37.119 [initandlisten] preallocating a journal file /srv/mongodb/journal/prealloc.2  Tue Jun  3 20:41:40.093 [initandlisten]         File Preallocator Progress: 1006632960/1073741824   93%  Tue Jun  3 20:41:52.450 [FileAllocator] allocating new datafile /srv/mongodb/local.ns, fillingwith zeroes…  Tue Jun  3 20:41:52.450 [FileAllocator] creating directory /srv/mongodb/\_tmp  Tue Jun  3 20:41:52.503 [FileAllocator] done allocating datafile /srv/mongodb/local.ns, size: 16MB,  took0.022 secs  Tue Jun  3 20:41:52.517 [FileAllocator] allocating new datafile /srv/mongodb/local.0, filling with zeroes…  Tue Jun  3 20:41:52.537 [FileAllocator] done allocating datafile /srv/mongodb/local.0, size: 64MB,  took0.02 secs  Tue Jun  3 20:41:52.538 [websvr] admin web console waiting for connections on port 28017  Tue Jun  3 20:41:52.538 [initandlisten] waiting for connections on port 27017 |

**Q206) install mongodb-10gen failed with apt-get**

Install mongodb-10gen as

https://docs.mongodb.org/manual/tutorial/install-mongodb-on-debian/but got error below:

<br>dpkg: error processing /var/cache/apt/archives/mongodb-10gen\_2.2.0\_amd64.deb (–unpack):

trying to overwrite ‘/usr/bin/mongoimport’, which is also in package mongodb-clients 1:1.4.4-3  
configured to not write apport reports  
dpkg-deb: subprocess paste killed by signal (Broken pipe)  
Errors were encountered while processing:

|  |  |
| --- | --- |
| 1 | /var/cache/apt/archives/mongodb-10gen\_2.2.0\_amd64.deb |

E: Sub-process /usr/bin/dpkg returned an error code (1)  
A bug here https://jira.mongodb.org/browse/SERVER-6910  
apt-get remove mongodb-clients first, and then the installation of mongodb-10gen should work.

**Q207) Mongodb Explain for Aggregation framework**

Is there an explain function for the Aggregation framework in MongoDB? I can’t see it in the documentation.  
If not is there some other way to check, how a query performs within the aggregation framework?  
I know with find you just do  
db.collection.find().explain()  
But with the aggregation framework I get an error

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14 | Skip code block  db.collection.aggregate(  { $project : { “Tags.\_id” : 1 }},  { $unwind : “$Tags” },  { $match: {$or: [{“Tags.\_id”:”tag1?},{“Tags.\_id”:”tag2?}]}},  {  $group:  {  \_id : { id: “$\_id”},  “count”: { $sum:1 }  }  },  { $sort: {“count”:-1}}  ).explain() |

There isn’t an explain() feature for the Aggregation Framework as at MongoDB 2.2.0, however you should read up on the documentation regarding Optimizing Performance.  
The following pipeline operators take advantage of an index when they occur at the beginning of the pipeline:

|  |  |
| --- | --- |
| 1 | $match $sort $limit $skip |

The above operators can also use an index when placed before the following aggregation operators:  
$project $unwind $group  
So for your specific example, the pipeline will execute faster if you add an initial $match at the beginning to reduce the number of documents that need to be projected/unwound for the subsequent$match.  
Update  
MongoDB 2.6 (which is currently a Release Candidate) now includes an explain option for the aggregation pipeline.

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15 | Skip code block  db.collection.aggregate([  { $project : { “Tags.\_id” : 1 }},  { $unwind : “$Tags” },  { $match: {$or: [{“Tags.\_id”:”tag1?},{“Tags.\_id”:”tag2?}]}},  { $group: {  \_id : “$\_id”,  count: { $sum:1 }  }},  {$sort: {“count”:-1}}  ],  {  explain:true  }  ) |

Starting with version 2.6.x mongodb allows users to do explain with aggregation framework.  
All you need to do is to add explain : true

|  |  |
| --- | --- |
| 1  2  3  4 | db.records.aggregate(  [ …your pipeline…],  { explain: true }  ) |

Thanks to Rafa, I know that it was possible to do even in 2.4, but only through runCommand(). But now you can use aggregate as well.

**Related Page:**[**MongoDB Aggregate**](https://mindmajix.com/mongodb-aggregate)

**Q208) Run script against replica set in MongoDB**

I have replica set of 3 nodes and I want to run a cleanup script against it every end of day. What I would do if there was only single node would be a simple bash script:  
~/mongo/bin/mongo host:port cleanupScript.js  
But since I want to run the same script against replica-set I can’t use this approach. I would need to somehow find which node is primary and run the script against that node.  
So the question: Is there a way how to run the script against whole replica set and let the mongoprocess pick the primary node and execute on it?  
Thanks!  
The mongo shell can connect directly to a replica set – this works with 2.4 (current), 2.2 (previous) and 2.0 (the version before that).  
Assuming you have a replica set called myrs and your hosts are host1:27017, etc. use the following syntax:  
mongo -host myrs/host1:27017  
The shell will figure out the rest, including connecting to the primary and if the primary steps down or goes away it will reconnect to the new primary once it’s elected.  
Unfortunately, the ‘mongo’ shell does not support making connections to a replica set: only to individual nodes. You have two choices if you want to do this:  
1. Use a different language, which supports making a connection to a replica set. (PHP, Python, Perl, Java and Ruby all support this.)  
2. Have a driver script that runs an ‘rs.status()’ command, parses the output of that command, and determines the current primary. If it’s not the node you’ve connected to, then re-connect to the correct primary node  
I wish I had a better answer for you.  
I normally set priority for a node in replicaset. This gives me freedom to chose which node should get read and write load.  
In your case, I think, if you set priority for the nodes then you can always run your script against the highest priority node as that node will be primary almost all the time.  
Setting priority is quite simple and straight forward. You can check this link https://docs.mongodb.org/manual/tutorial/force-member-to-be-primary/  
I hope this will solve your problem.  
OKAY…. Probably this is what you need..

|  |  |
| --- | --- |
| 1  2  3  4 | #!/bin/bash  PRIMARY=`~/mongo/bin/mongo localhost:27017 –eval “printjson(rs.isMaster())” | grep “primary” | cut -d””” -f4`  echo “$PRIMARY”  ~/mongo/bin/mongo “$PRIMARY” cleanupScript.js |

Run this on any node and it will give you the “server:port” of the primary. Give full path to the mongo executable.. just to avoid errors.  
PS: the whole command is in the backticks. Somehow those are not visible so thought of telling you.

**Q209) Couldn’t connect to server 127.0.0.1:27017**

I’m getting the following error:

|  |  |
| --- | --- |
| 1  2  3  4  5  6 | alex@alex-K43U:/$ mongo  MongoDB shell version: 2.2.0  connecting to: test  Thu Oct 11 11:46:53 Error: couldn’t connect to server 127.0.0.1:27017 src/mongo/shell/mongo.js:91  exception: connect failed  alex@alex-K43U:/$ |

This is what happens when I try to start mongodb:  
\* Starting database mongodb                                             [fail]  
I already tried mongo –repair  
I made chown and chmod to var, lib, and data/db and log mongodb.  
Not sure what else to do. Any suggestions?  
**mongodb.log:**  
Skip code block

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25  26  27  28  29  30  31  32  33  34  35  36  37  38  39  40  41  42  43  44  45  46  47  48  49  50  51  52  53  54  55  56  57  58 | \*\*\*\*\* SERVER RESTARTED \*\*\*\*\*  Thu Oct 11 08:29:40  Thu Oct 11 08:29:40 warning: 32-bit servers don’t have journaling enabled by default. Please use –journal if you want durability.  Thu Oct 11 08:29:40  Thu Oct 11 08:29:41 [initandlisten] MongoDB starting : pid=1052 port=27017 dbpath=/var/lib/mongodb 32-bit host=alex-K43U  Thu Oct 11 08:29:41 [initandlisten]  Thu Oct 11 08:29:41 [initandlisten] \*\* NOTE: when using MongoDB 32 bit, you are limited to about 2 gigabytes of data  Thu Oct 11 08:29:41 [initandlisten] \*\*       see https://blog.mongodb.org/post/137788967/32-bit-limitations  Thu Oct 11 08:29:41 [initandlisten] \*\*       with –journal, the limit is lower  Thu Oct 11 08:29:41 [initandlisten]  Thu Oct 11 08:29:41 [initandlisten] db version v2.2.0, pdfile version 4.5  Thu Oct 11 08:29:41 [initandlisten] git version: f5e83eae9cfbec7fb7a071321928f00d1b0c5207  Thu Oct 11 08:29:41 [initandlisten] build info: Linux domU-12-31-39-01-70-B4 2.6.21.7-2.fc8xen #1 SMP Fri Feb 15 12:39:36 EST 2008 i686 BOOST\_LIB\_VERSION=1\_49  Thu Oct 11 08:29:41 [initandlisten] options: { config: “/etc/mongodb.conf”, dbpath: “/var/lib/mongodb”, logappend: “true”, logpath: “/var/log/mongodb/mongodb.log” }  Thu Oct 11 08:29:41 [initandlisten] Unable to check for journal files due to: boost::filesystem::basic\_directory\_iterator constructor: No such file or directory: “/var/lib/mongodb/journal”  \*\*\*\*\*\*\*\*\*\*\*\*\*\*  Unclean shutdown detected.  Please visit https://dochub.mongodb.org/core/repair for recovery instructions.  \*\*\*\*\*\*\*\*\*\*\*\*\*  Thu Oct 11 08:29:41 [initandlisten] exception in initAndListen: 12596 old lock file, terminating  Thu Oct 11 08:29:41 dbexit:  Thu Oct 11 08:29:41 [initandlisten] shutdown: going to close listening sockets…  Thu Oct 11 08:29:41 [initandlisten] shutdown: going to flush diaglog…  Thu Oct 11 08:29:41 [initandlisten] shutdown: going to close sockets…  Thu Oct 11 08:29:41 [initandlisten] shutdown: waiting for fs preallocator…  Thu Oct 11 08:29:41 [initandlisten] shutdown: closing all files…  Thu Oct 11 08:29:41 [initandlisten] closeAllFiles() finished  Thu Oct 11 08:29:41 dbexit: really exiting now  EDIT:  I removed the lock then did mongod repair and got this error:  Thu Oct 11 12:05:37 [initandlisten] exception in initAndListen: 10309 Unable to create/openlock file: /data/db/mongod.lock errno:13 Permission denied Is a mongod instance already running?, terminating  so I did it with sudo:  Skip code block  alex@alex-K43U:~$ sudo mongod –repair  Thu Oct 11 12:05:42  Thu Oct 11 12:05:42 warning: 32-bit servers don’t have journaling enabled by default. Please use –journal if you want durability.  Thu Oct 11 12:05:42  Thu Oct 11 12:05:42 [initandlisten] MongoDB starting : pid=5129 port=27017 dbpath=/data/db/ 32-bit host=alex-K43U  Thu Oct 11 12:05:42 [initandlisten]  Thu Oct 11 12:05:42 [initandlisten] \*\* NOTE: when using MongoDB 32 bit, you are limited to about 2 gigabytes of data  Thu Oct 11 12:05:42 [initandlisten] \*\*       see https://blog.mongodb.org/post/137788967/32-bit-limitations  Thu Oct 11 12:05:42 [initandlisten] \*\*       with –journal, the limit is lower  Thu Oct 11 12:05:42 [initandlisten]  Thu Oct 11 12:05:42 [initandlisten] db version v2.2.0, pdfile version 4.5  Thu Oct 11 12:05:42 [initandlisten] git version: f5e83eae9cfbec7fb7a071321928f00d1b0c5207  Thu Oct 11 12:05:42 [initandlisten] build info: Linux domU-12-31-39-01-70-B4 2.6.21.7-2.fc8xen #1 SMP Fri Feb 15 12:39:36 EST 2008 i686 BOOST\_LIB\_VERSION=1\_49  Thu Oct 11 12:05:42 [initandlisten] options: { repair: true }  Thu Oct 11 12:05:42 [initandlisten] Unable to check for journal files due to: boost::filesystem::basic\_directory\_iterator constructor: No such file or directory: “/data/db/journal”  Thu Oct 11 12:05:42 [initandlisten] finished checking dbs  Thu Oct 11 12:05:42 dbexit:  Thu Oct 11 12:05:42 [initandlisten] shutdown: going to close listening sockets…  Thu Oct 11 12:05:42 [initandlisten] shutdown: going to flush diaglog…  Thu Oct 11 12:05:42 [initandlisten] shutdown: going to close sockets…  Thu Oct 11 12:05:42 [initandlisten] shutdown: waiting for fs preallocator…  Thu Oct 11 12:05:42 [initandlisten] shutdown: closing all files…  Thu Oct 11 12:05:42 [initandlisten] closeAllFiles() finished  Thu Oct 11 12:05:42 [initandlisten] shutdown: removing fs lock…  Thu Oct 11 12:05:42 dbexit: really exiting now |

But still having the same problem.  
The log indicates that mongodb is terminating because there is an old lock file.  
If you are not and were not running with journaling, remove the lock file, run repair, and start mongodb again.  
If you are or were running with journaling turned on, see the relevant Mongo DB docs. Note that they say “If you are running with Journaling you should not do a repair to recover to a consistent state.” So if you were journaling, the repair may have made things worse.  
Skip code block  
**Step 1:** Remove lock file.  
sudo rm /var/lib/mongodb/mongod.lock  
**Step 2:** Repair mongodb.  
mongod –repair  
**Step 3:** start mongodb.  
sudo start mongodb  
or  
sudo service mongodb start  
**Step 4**: Check status of mongodb.  
sudo status mongodb  
or  
sudo service mongodb status  
**Step 5:** Start mongo console.  
mongo  
Did you run mongod before running mongo?  
I followed installation instructions for mongodb from https://docs.mongodb.org/manual/tutorial/install-mongodb-on-os-x/ and I had the same error as you only when I ran mongo before actually running the mongo process with mongod. I thought installing mongodb would also launch it but you need to launch it manually with mongod before you do anything else that needs mongodb.  
try  
sudo service mongodb start  
I solved my problem by this

**Q210) Operate on all databases from the mongo shell**

We have a system with many different mongo databases. I regularly want to write ad-hoc queries that will apply to all (or a subset) of them, without having a priori knowledge of what databases are there.  
I can do show dbs, which will visually print a list, but is there a way to do something like:

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25 | var db\_list = listDatabases();  for (i = 0; i < db\_list.length; i++) {  do\_something(db\_list[i])  }  My problem with show dbs is that it doesn’t capture any return values, so I can’t do anything productive with the output.  You can use the ‘listDatabases’ admin command for that:  var db\_list = db.adminCommand(‘listDatabases’);  That returns an object that looks like this:  Skip code block  {  “databases” : [  {  “name” : “test”,  “sizeOnDisk” : 2097152000,  “empty” : false  },  {  “name” : “local”,  “sizeOnDisk” : 1,  “empty” : true  }  ],  “totalSize” : 8487174144,  “ok” : 1  } |

**Q211) I accidentally named a collection “stats” in MongoDB and now cannot rename it**

Oops.  
I use mongoose, and accidentally created a collection “stats”. I didn’t realize this was going to be an issue until a few weeks later, so I now need to rename (rather than just delete) the collection…  
However, my attempts have hit a predictable problem:

|  |  |
| --- | --- |
| 1 | PRIMARY> db.stats.find(); |

Thu Oct 18 10:39:43 TypeError: db.stats.find is not a function (shell):1

|  |  |
| --- | --- |
| 1  2  3  4 | PRIMARY> db.stats.renameCollection(‘statssnapshots’);  Thu Oct 18 10:39:45 TypeError: db.stats.renameCollection is not a function (shell):1  Try db[“stats”].find() and db[“stats”].renameCollection(‘statssnapshots’).  You could also do db.getCollection(“stats”).find(). |

**Q212) Set max on mongo capped collection**

I have an existing collection that I need to convert into a Capped Collection using the method listed:

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9 | > db.runCommand({“convertToCapped”: “mycoll”, size: 100000});  However, the max field is not accepted as a parameter  > db.mycol1.convertToCapped  function (bytes) {  if (!bytes) {  throw “have to specify # of bytes”;  }  return this.\_dbCommand({convertToCapped: this.\_shortName, size: bytes});  } |

Any idea how to set this?  
max is only an option in the createCollection method, not convertToCapped:  
db.createCollection(“mycoll”, {capped:true, size:100000, max:100});  
There’s a cloneCollectionAsCapped, but it doesn’t look like there’s a max doc option there either:

<strong>https://docs.mongodb.org/manual/reference/command/cloneCollectionAsCapped/</strong>

You may need to create a new capped collection with the max parameter and transfer data and indices from the existing collection. See https://learnmongo.com/posts/easily-move-documents-between-collections-or-databases/

**Q213) Does MongoDB reuse deleted space?**

First off, I know about this question:  
Auto compact the deleted space in mongodb?  
My question is not about shrinking DB file sizes though, but more about the reuse of deleted space. Say I have 100K documents in a collection, I then delete 50K of those. Will Mongo reuse the space within its data file that the deleted documents have freed? Or are they simply “marked” as deleted?  
I don’t care so much about the actual size of the file on disk, its more about “does it just grow and grow”.  
When documents are deleted the space left behind is put into a free list, yes. However, to use the space there will need to be similarly sized documents inserted later, and MongoDB will need to find an appropriate space for that document within a certain time frame (once it times out looking at the list, it will just append) otherwise the space re-use is not going to happen very often. This deletion is done within the data files, so there is no disk space reclamation happening here – all of this is done internally within the existing data files.  
If you subsequently do a repair, or resync a secondary from scratch, the data files are rewritten and the space on disk will be reclaimed (any padding on docs is also removed). This is where you will see actual space reclamation on-disk. For any other actions (compact included) the on disk usage will not change and may even increase.  
With 2.2 you can now use the collMod command and the use PowersOf2Sizes option to make the re-use of deleted space more likely. This means that the initial space allocation for a document is a bit less efficient (512 bytes for a 400 byte doc for example) but means that when a new doc is inserted it is more likely to be able to re-use that space. If you are deleting (or growing and hence moving) documents a lot, then this will be more efficient in the long term.  
For anyone that is interested, one of the people that wrote a lot of the storage code (Mathias Stearn) has a great presentation about the storage internals, which can be found here

**Q214) Can not delete collection from mongodb**

Can not delete the collection from the shell,  
The thing that the collection is available and my php script is accessing it (selecting|updating)  
But when I used:  
db.\_registration.drop()  
it gives me an error:  
Date, JS Error: TypeErrorL db.\_registration has no properties (shell): 1  
The problem is not with deleting the collection. The problem is with accessing the collection. So you would not be able to update, find or do anything with it from the shell. As it was pointed in mongodb JIRA, this is a bug when a collection has characters like \_, – or .  
Nevertheless this type of names for collections is acceptable, but it cause a problem in shell.  
You can delete it in shell with this command:

|  |  |
| --- | --- |
| 1 | db.getCollection(“\_registration”).drop() |

but I would rather rename it (of course if it is possible and will not end up with a lot of changing).

**Q215) explain() in Mongodb: differences between “nscanned” and “nscannedObjects”**

I cannot get the exact difference between “nscanned” and “nscannedObjects” in the Mongodb’s explain query output.  
On MongoDB Explain documentation I can read:  
**nscanned**Number of items (documents or index entries) examined. Items might be objects or index keys. If a “covered index” is involved, nscanned may be higher than nscannedObjects.  
**nscannedObjects** Number of documents scanned.  
What’s the different between these two fields? And more specific what does exactly mean when I have a query, which uses a BtreeCursor (an index), and these two fields have two different

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9 | values, for example:  {  “cursor” : “BtreeCursor a\_1\_b\_1”,  “isMultiKey” : false,  “n” : 5,  “nscannedObjects” : 5,  “nscanned” : 9,  (…)  } |

I know what a “covered index” is. I would like to understand exactly what the query did in the example above. Did it pass through (“scanned”) 9 elements (nscanned = 9), where all of them are index entries and read (“examined”) the value of only 5 of them (nscannedObjects = 5) to produce the result set?  
This means that :  
The query returned 5 documents – n  
scanned 9 documents from the index – nscanned  
and then read 5 full documents from the collection – nscannedObjects  
Similar example is given at :

<br><strong>https://docs.mongodb.org/manual/core/read-operations/#measuring-index-use</strong>

**Q216) MongoDb Database vs Collection**

I am designing a system with MongoDb (64 bit version) to handle a large amount of users (around 100,000) and each user will have large amounts of data (around 1 million records).  
What is the best strategy of design?  
Dump all records in single collection  
Have a collection for each user  
Have a database for each user.  
Many Thanks,  
So you’re looking at somewhere in the region of 100 billion records (1 million records \* 100,000 users).  
The preferred way to deal with large amounts of data is to create a sharded cluster that splits the data out over several servers that are presented as single logical unit via the mongo client.  
Therefore the answer to your question is put all your records in a single sharded collection.  
The number of shards required and configuration of the cluster is related to the size of the data and other factors such as the quantity and distribution of reads and writes. The answers to those questions are probably very specific to your unique situation, so I won’t attempt to guess them.  
I’d probably start by deciding how many shards you have the time and machines available to set up and testing the system on a cluster of that many machines. Based on the performance of that, you can decide whether you need more or fewer shards in your cluster

**Q217) MongoDB upsert – insert or update**

I can’t find this in the documentation in any of the obvious places. I’d like to know if it is possible to know if mongo executed an insert or update in the upsert operation? Thanks.  
Yes there is, on a safe call (or getLastError) the update function will return an array with an upsert field and a updatedExisting field.  
You can read the PHP version of this here:

<strong>https://php.net/manual/en/mongocollection.insert.phptowards</strong>

the bottom.  
As it says within the documentation on upserted:  
If an upsert occured, this field will contain the new record’s \_id field. For upserts, either this field or updatedExisting will be present (unless an error occurred).  
So upserted contains the \_id of the new record if a insert was done or it will incrementupdatedExisting if it updated a record.  
I am sure a similar thing appears in all drivers.  
**Edit**  
It will actually be a boolean in the updatedExisting field of true or false  
For reference only, in node.js:

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20 | collection.update( source, target, { upsert: true }, function(err, result, upserted) {  …  });  For reference only, in node.js using Mongoose 3.6:  model.update( findquery, updatequery, { upsert: true }, function(err, numberAffected, rawResponse) {  …  });  Where rawResponse looks like this when it has updated an existing document:  { updatedExisting: true,  n: 1,  connectionId: 222,  err: null,  ok: 1 }  And it looks like this when it has created a new document:  { updatedExisting: false,  upserted: 51eebc080eb3e2208a630d8e,  n: 1,  connectionId: 222,  err: null,  (Both cases would return numberAffected = 1) |

**Q218) How to insert an element to MongoDB internal list?**

I have the following doc stored in MongoDB:

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20 | Skip code block  {  name: ‘myDoc’,  list: [  {  id:1  items:[  {id:1, name:’item1'},  {id:2, name:’item2'}  ]  },  {  id:2  items:[  {id:1, name:’item1'},  {id:3, name:’item3'}  ]  }  ]  } |

I found a way to add an element to ‘list’ using ‘$addToSet’ but I couldn’t find a way to add to a specific list of ‘items’ an item.  
e.g. I get the following:

|  |  |
| --- | --- |
| 1 | {id:5, name:’item5'} |

and I want to add it to the element’s item in the list with id:2.  
One way of doing it would be with $push:

|  |  |
| --- | --- |
| 1  2  3  4 | db.col.update(  { name: ‘doc’, ‘list.id’: 2 },  {$push: {‘list.$.items’: {id: 5, name: ‘item5’}}}  ) |

<br>https://docs.mongodb.org/manual/reference/operator/push/

You can also replace $push with other operators like (possibly) $addToSet to get the exact results you are looking for.  
You can use this: –

|  |  |
| --- | --- |
| 1  2  3  4  5 | > var toInsert = {id: 5, name: ‘item6’}  > db.abc.update(  { name: ‘myDoc’, list: { $elemMatch: { id: 2 } } },  { $addToSet: { ‘list.$.items’: toInsert } }  ) |

The query part will find the document from the list array with id = 2. Then we use $ positional element to add a new element at that array index.  
See positional $ operator  
You can also replace list: {$elemMatch: {id: 2}} with just ‘list.id’: 2.  
But using $elemMatch will be better, when you want to update based on multiple elements of your array. For e.g., when your matching criteria is id and some another field in the array say length: –

<br>list: {$elemMatch: {id: 2, length: 5}}

**Q219) Fastest way to remove duplicate documents in mongodb**

I have approximately 1.7M documents in mongodb (in future 10m+). Some of them represent duplicate entry which I do not want. Structure of document is something like this:

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8 | {  \_id: 14124412,  nodes: [  12345,  54321  ],  name: “Some beauty”  } |

Document is duplicate if it has at**least one node same** as another document with **same name**. What is the fastest way to remove duplicates?  
Assuming you want to permanently delete docs that contain a duplicate name + nodes entry from the collection, you can add a unique index with the dropDups: true option:  
db.test.ensureIndex({name: 1, nodes: 1}, {unique: true, dropDups: true})  
As the docs say, use extreme caution with this as it will delete data from your database. Back up your database first in case it doesn’t do exactly as you’re expecting.

**Related Page:**[**MongoDB Update Document**](https://mindmajix.com/mongodb-update-document)

**Q220) MongoDB $query operator ignores index?**

I created a test MongoDB collection “sampleCollection” with documents which looks like:

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25  26  27 | “\_id” : ObjectId(“510929e041cb2179b41ace1c”),  “stringField” : “Random string0”,  “longField” : NumberLong(886)  and has index on field “stringField”. When I execute  db.sampleCollection.find({“stringField”:”Random string0?}).explain()  everything is ok:  Skip code block  “cursor” : “BtreeCursor stringField\_1”,  “isMultiKey” : false,  “n” : 2,  “nscannedObjects” : 2,  “nscanned” : 2,  “nscannedObjectsAllPlans” : 2,  “nscannedAllPlans” : 2,  “scanAndOrder” : false,  “indexOnly” : false,  “nYields” : 0,  “nChunkSkips” : 0,  “millis” : 0,  “indexBounds” : {  “stringField” : [  [  “Random string0”,  “Random string0”  ]  ]  } |

but

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17 | db.sampleCollection.find({$query:{“stringField”:”Random string0?}}).explain()  gets me  Skip code block  “cursor” : “BasicCursor”,  “isMultiKey” : false,  “n” : 0,  “nscannedObjects” : 4,  “nscanned” : 4,  “nscannedObjectsAllPlans” : 4,  “nscannedAllPlans” : 4,  “scanAndOrder” : false,  “indexOnly” : false,  “nYields” : 0,  “nChunkSkips” : 0,  “millis” : 0,  “indexBounds” : {  } |

This is not looks like a problem, but I’m using **org.springframework.data.mongodb**framework in production and usage of query constructions is an only way to write repositories. And thus I have a db which completely ignores indexed data.  
Is it correct? Or I misunderstood something?  
That was funny i cannot decide to say it is a bug or not it is up to you:  
There are two available syntax: https://docs.mongodb.org/manual/reference/operator/query/  
When you using:

|  |  |
| --- | --- |
| 1 | db.collection.find( { age : 25 } ) |

also will

|  |  |
| --- | --- |
| 1  2 | db.collection.find( { age : 25 } ).explain()  db.collection.find( { age : 25 } ).hint(someindex) |

work fine.  
When you using your solution (the other syntax):

|  |  |
| --- | --- |
| 1 | db.collection.find( { $query: { age : 25 } } ) |

the output of  
db.sampleCollection.find({$query:{“stringField”:”Random string0?}}).explain()  
Will show like the query not using the index  
if you also use .hint for the index it will omit the result. :) (That is i do not really understand)  
Fortunately there is another syntax for these operations too: you can use:  
db.sampleCollection.find({$query:{“stringField”:”Random string0?}, $explain:1})  
it will have the right output and showed for me the usage of the index. Also there is similar syntax for $hint.  
You can check the documentation here: https://docs.mongodb.org/manual/reference/meta-query-operators/  
I found this really interesting so i turned on the profiler:  
i made a test collection (queryTst) with around 250k docs each with only \_id and an age field in the structure with an index on age.  
For this query:

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25  26  27  28  29  30  31  32  33  34  35  36  37  38  39  40  41  42  43  44  45  46  47  48  49  50  51  52  53  54  55  56  57  58  59  60  61  62  63  64  65  66  67  68  69  70  71  72  73  74  75  76  77  78  79  80  81  82  83  84  85  86  87  88  89  90  91  92  93  94  95  96  97  98  99  100  101  102  103  104  105  106  107  108  109  110  111  112  113  114  115  116  117  118  119  120  121  122  123  124  125  126  127  128  129  130  131  132  133  134  135  136  137  138  139  140 | db.queryTst.find({$query:{“age”:16},$explain:1})  i got:  Skip code block  {  “cursor” : “BtreeCursor age\_1”,  “isMultiKey” : false,  “n” : 2,  “nscannedObjects” : 2,  “nscanned” : 2,  “nscannedObjectsAllPlans” : 2,  “nscannedAllPlans” : 2,  “scanAndOrder” : false,  “indexOnly” : false,  “nYields” : 0,  “nChunkSkips” : 0,  “millis” : 0,  “indexBounds” : {  “age” : [  [  16,  16  ]  ]  },  “allPlans” : [  {  “cursor” : “BtreeCursor age\_1”,  “n” : 2,  “nscannedObjects” : 2,  “nscanned” : 2,  “indexBounds” : {  “age” : [  [  16,  16  ]  ]  }  }  ],  “oldPlan” : {  “cursor” : “BtreeCursor age\_1”,  “indexBounds” : {  “age” : [  [  16,  16  ]  ]  }  },  “server” : “”  }  for this:  db.queryTst.find({$query:{“age”:16},$explain:1}).explain()  i got:  Skip code block  “cursor” : “BasicCursor”,  “isMultiKey” : false,  “n” : 0,  “nscannedObjects” : 250011,  “nscanned” : 250011,  “nscannedObjectsAllPlans” : 250011,  “nscannedAllPlans” : 250011,  “scanAndOrder” : false,  “indexOnly” : false,  “nYields” : 0,  “nChunkSkips” : 0,  “millis” : 103,  “indexBounds” : {  },  in the profiler log: for the first  Skip code block  {  “ts” : ISODate(“2013-01-30T20:35:40.526Z”),  “op” : “query”,  “ns” : “test.queryTst”,  “query” : {  “$query” : {  “age” : 16  },  “$explain” : 1  },  “ntoreturn” : 0,  “ntoskip” : 0,  “nscanned” : 2,  “keyUpdates” : 0,  “numYield” : 0,  “lockStats” : {  “timeLockedMicros” : {  “r” : NumberLong(368),  “w” : NumberLong(0)  },  “timeAcquiringMicros” : {  “r” : NumberLong(8),  “w” : NumberLong(5)  }  },  “nreturned” : 1,  “responseLength” : 567,  “millis” : 0,  “client” : “127.0.0.1”,  “user” : “”  }  for the second:  Skip code block  {  “ts” : ISODate(“2013-01-30T20:35:47.715Z”),  “op” : “query”,  “ns” : “test.queryTst”,  “query” : {  “query” : {  “$query” : {  “age” : 16  },  “$explain” : 1  },  “$explain” : true  },  “ntoreturn” : 0,  “ntoskip” : 0,  “nscanned” : 250011,  “keyUpdates” : 0,  “numYield” : 0,  “lockStats” : {  “timeLockedMicros” : {  “r” : NumberLong(104092),  “w” : NumberLong(0)  },  “timeAcquiringMicros” : {  “r” : NumberLong(13),  “w” : NumberLong(5)  }  },  “nreturned” : 1,  “responseLength” : 373,  “millis” : 104,  “client” : “127.0.0.1”,  “user” : “”  } |

which somehow means to me that is the explain() cause the table scan in the mixed syntax.  
this is not a bug, I happened to have the same question you had, and was about to report it as an issue in mongodb, but reading the documentation it specifies that you shouldn’t mix cursor methods ( like the explain() ) with this query format, you should use $explain, like in the example provided in this page:

https://docs.mongodb.org/manual/reference/operator/query/

**Q223). How to index an $or query with sort**

Suppose I have a query that looks something like this:  
Skip code block

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13 | db.things.find({  deleted: false,  type: ‘thing’,  $or: [{  ‘creator.\_id’: someid  }, {  ‘parent.\_id’: someid  }, {  ‘somerelation.\_id’: someid  }]  }).sort({  ‘date.created’: -1  }) |

That is, I want to find documents that meets one of those three conditions and sort it by newest. However, $or queries do not use indexes in parallel when used with a sort. Thus, how would I index this query?

<br><strong>https://docs.mongodb.org/manual/core/indexes/#index-behaviors-and-limitations</strong>

You can assume the following selectivity:  
1. deleted- 99%  
2. type- 25%  
3. creator.\_id,parent.\_id, somerelation.\_id – < 1%  
Now you are going to need more than one index for this query, there is no doubt about that.  
The question is what indexes?  
Now you have to take into consideration that none of your $ors will be able to sort their data cardinally in a optimal manner using the index due to a bug in MongoDBs query optimiser: https://jira.mongodb.org/browse/SERVER-1205.  
So you know that the $or will have some performance problems with a sort and that putting the sort field into the $or clause indexes is useless atm.  
So condering this the first index you want is one that covers the base query you are making. As @Leonid said you can make this into a compound index, however I would not do it the order he has done it. Instead I would do:

|  |  |
| --- | --- |
| 1 | db.col.ensureIndex({type:-1,deleted:-1,date.created:-1}) |

I am very unsure about the deleted field being in the index at all due to it’s super low selectivity, it could in fact create a less performant operation (this is true for most databases including SQL) being in the index rather than being taken out. This part will need testing by you, maybe the field should be last (?)

Q1) Compare MongoDB and Cassandra

MongoDB:

* data mode are document
* database scalability is read only
* query of the data is multi-index

Cassandra:

* data mode are big table like
* database scalability is write only
* query of the data is using key or scana

Q2) what make magodb best?

it is consider has nosql database  because documented oriented (do).

Q3) When do we use Namespace in MongoDB?

in  database hte namespace is used

Q4) when an index does not fit into RAM?

index is very huge then index will not fit to RAM.

Q5) If you remove an object attribute, is it deleted from the database?

yes when the object attribute is delete then the object will be drop

Q6) Define MongoDB.

it th DO database which use in high availability and it is dynamic schema loction

Q7) What are the key features of mongodb?

* high perform
* high availability
* automatic scaling,

Q8) what is meant by CRUD?

create read update delete

Q9)What is sharing in MongoDB?

the process of storing the record in multiple Machine is know has sharing.

Q10) How can you see the connection used by Mongos?

Mongos use db\_adminCommand (“connPoolStats”);

Q11) Explain what is a replica set?

A replica set is the group of magodb instance which is host in same data set

Q12) How replication works in MongoDB?

the process of synchronization across the multiple server os know has replication.

Q13) While creating Schema in MongoDB what are the points need to be taken in consideration?

below are the following point need to be take care while carating the schema the combine objects into one document if you use them together for most frequent use cases optimize your schema

Q14) What is the syntax to create a collection?

syntax.

db.create Collection(name,options)

Q15) What is the syntax to drop a collection?

syntax:

drop collection in MongoDB is db.collection.drop()

[*Get MongoDB Online Training*](https://www.gangboard.com/database-training/mongodb-training?utm_source=GB_blog)

Q16) Mention what is Object ld composed of?

* Timestamp
* Client machine ID
* Client process ID
* byte incremented counter

Q17) Mention what is the command syntax for inserting a document?

syntax :

database.collection.insert (document).

Q18) What is the command syntax that tells you whether you are on the master server or not? And how many master does MongoDB allow?

* Command syntax
* isMaster()

Q19) Mention the command syntax that is used to view Mongo is using the link?

syntax

db.\_adminCommand(“connPoolStats.”)

Q20) Mention what is the basic syntax to use index in MongoDB?

syntax :

>db.COLLECTION\_NAME.ensureIndex ( {KEY:1} ).

Q21)Explain what is GridFS in MongoDB?

while  storing and retrieving large files such as images.

Q22) What are alternatives to MongoDB?

* Cassandr
* CouchDB
* Redis
* Riak

Q23) What kind of NoSQL database MongoDB is?

It is a document oriented database.

Q24) Which all languages can be used with MongoDB?

c,java,c++

Q25) How is MongoDB better than other SQL databases?

it Permits  highly flexible and scalable document structure. does not support such relationships.

Q26) Does MongoDB need a lot of RAM?

it can run on small amount of data because it is dynamic space are allocated and re-allocated.

Q27) What is Mongo DB?

It is an open source database and prominent NOSQL database.

Q28)     What are all the advantage of MongoDB?

* It is a Schema less.
* Structure of a single object is clear.
* No complex joins.
* Easy of scale-out

Q29) How to create the database in Mongo DB?

Use Database name

Example: Use MYDB

Q30) How to show the entire database?

Show dbs

Q31) What are the datatypes Mongo DB supports?

String, Integer, Boolean, Double, Array, Timestamp, Date, Regular expression

Q32) How to insert document in Mongo DB collection?

Db.dbname.insert(document)

Q33) How to display all the documents?

Db.dbname.find()

Q34) What is the purpose of pretty () method?

Pretty () method used to display in formatted way.

Q35) What is the use of limit ()?

It will used to limit the value display from the documents.

DB.DBNAME.find ().limit (number)

Q36) What are all the Aggregate function in Mongo DB?

Sum, Avg, Min, Max, Push, addToSet, first, last

Q37) What is replication and why we need?

It is the process of synchronizing data across multiple server. It provides redundancy and increases data availability with multiple copies of data on different database server.WHY: To keep data safe, High availability of data, disaster recovery, No downtime for maintenance.

Q38) What is the syntax of sort() method?

db.COLLECTION\_NAME.find().sort({KEY:1})

Q39) What is CRUD?

CRUD operations are Create, Read, Update, Delete.

Q40) What is the syntax to drop the collections?

Db.collection.drop()

[*Are You Interested in MongoDB Course ? Click here*](https://www.gangboard.com/database-training/mongodb-training?utm_source=GB_blog)

Q41) What is embedded documents?

Embedded documents are received while they were small, given the relations between the related data has been written in one body.

Q42) Define application level encryption.

Application Level Encryption provides encryption or per-per-written from the field in the application layer.

Q43) The encryption storage?

MongoDB encryption encrypts data stored on any storage or processing by operating system may not be able to access protected data.

Q44) When everything is in no way sufficient data to get extended to multi-cut?

That stands in a few MongoDB collection. Therefore, the list is kept in a lump of dough of all substances, it is either in size. It is only when there is additional time slot, there will be no piece of information was a few choices, but the lump is more than 1, data is far from a lot of slices and can be extended up to 64 MB.

Q45) The judge in the MongoDB and Couchbase and CouchbaseDB?

When the common segment DB DB is Couchbase Couchbase Brunfelsia necessary in many ways, crossing points, all of which differ in the form of ecological duplications etc. As to the execution

Q46) Miâ on how we use MongoDB?

By sequencing of database, and the names of the pages linked from a specified collection is used.

Q47) If you remove an object attribute database wipes?

So deleted. It is better to be safe to remove the attribute.

Q48) How can we move into the old directory file moveChunk?

Since, however, the present and the old files are converted to the backup file is moved at the time of the acquisition moveChunk directory space.

Q49) Explain the situation with the index does not fit in RAM?

MongoDB learned from the experts? Enroll today

Q50) What MongoDB will provide consistency?

The writer makes use of MongoDB is he that readeth, and the bolts of it, after allowing to access one considers arbitrary or that any one that my readers may at the same time as the collection of a database. And it always brings access to private and hated by others.

Q51) Why did not MongoDB 32-bit system because he chose?

32 bit candidate for a non-Brunfelsia DB is contained, that means running 32 bit with the MongoDB server, information and indexes require 2 GB. 32-bit, so it is not the only way to enter my thoughts.

Q52) How does working on Journaling MongoDB?

There is nothing memory Journaling saved. The book on-disk files are really useful to have said that the route habitual. DbPath interior is designed in the Journal subdirectory MongoDB.

Q53) How can we separate cursors connection with the write injustice?

Snapshot () method is used to separate them from coming in with cursors writes. Here is a list of query and make sure everyone comes to be, and the manner of his mother tongue, there has been only to look at the article.

Q54) Define MongoDB.

It may be a document is to be oriented database of speech, with which the top of their minds to ease of scalability and easy to get high performance. The Scheme supports dynamic policy.

Q55) Explain the image set.

I have to put the example of which is the group made up of Brunfelsia same knowledge of the set. To kill, and I need to provide high availability redundancy is the image of, and in the production of all of the deployments.

Q56) What are the key features of mongodb?

3 and of the sea There are, however, latae sententiae, which are mongodb scaling: High performance and high availability.

Q57) The Crud?

Mongodb provides Crud are created, Read, Update, Delete.

To learn more information about the MongoDB Crud click here.

Q58) The sharding?

Sharding to store data on multiple machines.

Q59) What is the sum of the MongoDB?

What matters are aggregations process data records and return the results of counting.

Q60) To define the nominal distance mongodb?

And what is the name of the collection database.

[*Become a MongoDB Certified Expert in 25Hours*](https://www.gangboard.com/database-training/mongodb-training?utm_source=GB_blog)

Q61)Syntax is used to create a collection mongodb?

db.createCollection (name, options) is used to create a collection mongodb.

Syntax collection.

Q62) That is to drop the mongodb?

db.collection.drop () is a collection to drop the mongodb.

Q63) Explain Infection.

Infection by more synchronous processing of data servers.

Q64) What is the use of index mongodb?

You can read a thing often used to provide a guideline for the high performance of queries.

Q65) The government should be inserted here because it is used?

database.collection.insert (document) is used for inserting the document.

Q66) What is the use of GridFS mongodb?

GridFS as I hear, is used for storage and for the most retrieving files, images, video files.

Q67) Journaling used?

Journaling is the use of the back of the mongodb to be saved.

Q68) That the government used to bond?

db\_adminCommand ( “connPoolStats”); See the connection is not used.

Q69)Define the image killing?

The first things to write replicas but embraced the culture.

Q70) Define secondary image killing.

Secondaries (page 565) Brand apply for the primary, and op oplog.

Q71) Select the MongoDB equivalent query for the below SQL query – INSERT INTO employee (eid, age, salary) VALUES (763357, 45, 56489.40);

employee.insert( { eid: 763357, age: 45, salary: 56489.40 } )

 Q72) Identify the **invalid** statement

employee.insertMany( { [ emp\_id:1, ename:”AAA” ] } )

 Q73) An “inventory” collection has documents stored in MongoDB in the following format:

{\_id:1,  status: “available”,items: [ { sku:”006548e8″, details: {……} }, { sku:”0067458e8″, details: {……}  }  ], stocklevel:35, seller: “aaa” }

 Which of the following commands can be used to find all the documents in the “inventory” collection having stocklevel from 20 to 50? [20 and 50 inclusive]

inventory.find( { $nor: [ { stocklevel: { $lt:20 } },{ stocklevel:{ $gt:50 } } ] } ), db. inventory.find ( { stocklevel : { $gte: 20, $lte: 50 } } )

 Q74) Given the following documents in the *electronics*collection:

{ \_id: 1, item: “oven”, price: 14000,quantity: 5 }

{ \_id: 2, item: “LCD”, price: 48000, quantity: 10 }

{ \_id: 3, item: “refrigerator”, price: 25000, quantity: 4 }

{ \_id: 4, item: “LCD”, price: 56000, quantity: 3 }

{ \_id: 5, item: “LCD”, price: 20000, quantity: 9 }

 What is the result of the below operation?

 db.electronics.update( {item : “LCD”, quantity : { $gte : 5 } },  { $set : { HD : “full HD” }, $inc : { price : 8000 } } )

Updates only the first matching document

 Q75) What would be the result of running the below query? – employee\_profiles.deleteOne(   {‘unit’: ‘ETA’})

Only the first matching document is deleted

 Q76) The aggregation query in MongoDB that will sort the collection with ename field ascending.

employee.aggregate( [ { $sort: {ename:1} } ] )

 Q77) The command that creates an index on the quantity field, descending

inventory.createIndex( { “stock.quantity”:-1 } )

Core MongoDB Interview Questions and Answers

Here is the list of frequently asked interview questions for MongoDB.

**Q1. How can db.ClockTime.update() convert these text-based values to a date datatype?**

**Ans.** There can be two solutions for the problem given over here:

* **First solution**
  1. > var cursor = db.ClockTime.**find**()
  2. > **while** (cursor.**hasNext**()) {
  3. … var doc = cursor.**next**();
  4. … db.ClockTime.**update**({\_id : doc.\_id}, {$set : {ClockInTime : new **Date**(doc.ClockInTime)}})
  5. … }
* **Second solution**
  1. db.ClockTime.**find**().**forEach**(**function**(doc) {
  2. doc.ClockInTime=new **Date**(doc.ClockInTime);
  3. db.ClockTime.**save**(doc);
  4. })

So using any of these you can convert text-based values to date datatype.

**Q2. How can you use map-reduce for grouping more than 10000 unique keys?**

**Ans.** For [**using MapReduce**](https://data-flair.training/blogs/hadoop-mapreduce-tutorial/) for grouping, use version 2.2. The db.collection.group() method’s returned array can contain up to 20,000 elements; that means at up to 20,000 unique groupings. For the group by operations that are having more than 20,000 unique groupings, MapReduce’s previous versions had a limit of 10,000 elements.

**Q3. In what way will you query live MongoDB data?**

**Ans.** By using the following utilities you can handle live MongoDB data.

* MongoHub is moved to a native Mac version.
* This is useful for tree and document views.
* **genghisapp** – This is a web-based GUI that is clean, light-weight, straight-forward, offers keyboard shortcuts, and works awesomely well. It also supports GridFS. It’s all a single script.

**Q4. How to update a specific key/value of an array field with MongoDB?**

**Ans.** For this let us suppose a collection content, as it contains articles, news, etc. These articles contain another array called the author which contains all the information about the author.

1. db.content.**update**({‘authors.abc’:’xyz’},{$set:{‘authors.$.address’:’Address That I wanted’}},false,true);

It updates all the records properly.

**Q.5 How to use the primary key in MongoDB?**

**Ans.** \_id field is reserved for the primary key in MongoDB, and that is a unique value. If you don’t set anything to \_id it will automatically fill it with “MongoDB Id Object”. But you can put any unique info in that field.

**Q6. How to delete everything from the MongoDB database?**

**Ans.** For [**deleting everything from the MongoDB database**](https://data-flair.training/blogs/mongodb-delete-document/), you can use the following code:

1. use [database];
2. db.**dropDatabase**();
3. Ruby code should be pretty similar.
4. Also, from the command line:
5. mongo [database] –eval “db.**dropDatabase**();”
6. Use
7. [databaseName]
8. db.Drop+**databaseName**();
9. drop collection
10. use databaseName
11. db.collectionName.**drop**();

**Q7. How to use $set in MongoDB?**

**Ans.** For a detailed explanation about how it works dynamically, I’ll give an example in PHP:

1. $action = **array**(“columns.$colNum.panels.$panelNum” => $newValue);

Yes there is the positional operator, but it does not appear to be advanced enough to change arrays within arrays, this might have changed in MongoDB 1.7.0

There is an alternative for this. Try to flatten it out. We can make a collection that contains panel & column objects:

1. column object:
2. {
3. \_id: // MongoId
4. type: ‘column’,
5. user: ‘username’,
6. order: 1,
7. width: 30,
8. }
9. panel object:
10. {
11. \_id: //MongoId
12. type: ‘panel’,
13. user: ‘username’,
14. parentColumn: //the columns \_id string
15. top: 125,
16. left: 100
17. }

Then find all columns that belong to a user:

1. **find**({ type: ‘column’, user:’username’});

We can detect all panels for a particular column by doing:

1. **find**({type: ‘panel’, columnOwner:’ownerID’});

Since each column and panel will contain a unique ID given by MongoDB to it, we can query easily and atomically set options.

1. **update**({‘\_id’: **ObjectId**(‘idstring’)}, {$set : { ‘top’ : 125}});

**Q8. Using what can you print more than 20 items in MongoDB’s shell?**

**Ans.**

1. db.example.**find**().**toArray**()
2. db.example.**find**().**forEach**(printjson)

It will both print out a very expanded view of each document instead of the 1-line version for find():

1. DBQuery.shellBatchSize = 300

**Q9. How to perform join operations in MongoDB?**

**Ans.** From MongoDB 3.2 onwards we can perform join operation. The new $lookup operator added with the aggregation pipeline is essentially similar to a left outer join. This is an example of the same:

1. {
2. $lookup:
3. {
4. from: <collection to join>,
5. localField: <field from the input documents>,
6. foreignField: <field from the documents of the "from" collection>,
7. as: <output array field>
8. }
9. }

**Q10. Explain oplog.**

**Ans.** The operational log(oplog) is a special type of capped collection that keeps a rolling record of all operations that modify the data stored in your databases. Firstly it applies database operations on the primary and then records these operations on the primary’s oplog. Then, the secondary members copy and implement these operations in an asynchronous process.

**Q11. When should the data be normalized in MongoDB?**

**Ans.** It depends on your goals. Normalization will provide an update efficient data representation. Denormalization will make data reading efficiently.

In general, use embedded data models (denormalization) when:

* When you have “contains” relationships between entities.
* When you have one-to-many relationships between entities. In these relationships, the “many” or child documents always display with or are seen in the context of the “one” or parent documents.

In general, use normalized data models:

* When embedding leads to duplication of data but will not give sufficient read performance benefits to outweigh the implications of the duplication.
* To represent more complex many-to-many relationships.
* To model large hierarchical data sets.

**Q12. Comparison between MongoDB and Cassandra.**

**Ans.**

**MongoDB**:

* Data mode are the document
* Database scalability is read-only
* Query of the data is multi-index

**Cassandra**:

* Data mode are a big table like
* Database scalability is write only
* Query of the data by using scan or key

[**Check out Cassandra vs MongoDB for the detailed description**](https://data-flair.training/blogs/cassandra-vs-mongodb/)

**Q13. What does ObjectId consist of?**

**Ans.** It consists of the following:

* Timestamp
* Client machine ID
* Client process ID
* Byte incremented counter

**Q14. Explain about replication and when should we use it?**

**Ans.** [**Replication is the process of synchronizing data across multiple servers**](https://data-flair.training/blogs/mongodb-replication-and-sharding/) so that data is not lost in any condition. It gives redundancy and rises data availability with many copies of data on other database servers. We can use it for various purposes like for keeping the data safe, for high availability of data, disaster recovery and in no downtime for maintenance.

**Q15. When and up to what extent does data get extended to multi-slice?**

**Ans.** The MongoDB scrap stands on a collection. So all substances are kept in a lump or mass. When there is an additional time slot only then there will be more than a few slice data achievement choices, but at the time, when there is more than 1 lump, data gets extend to many slices and it can further extend to 64 MB.

**Q16. Enlist the limitations of the 32-bit version of MongoDB.**

**Ans.** MongoDB uses memory-mapped files. When we run a 32-bit build of MongoDB, the total size of storage for the server, including indexes and data, is 2 gigabytes. Due to this, don’t deploy MongoDB to production on 32-bit machines.

And in the 64-bit build of MongoDB, there’s virtually no limit to storage size. For production deployments, we strongly recommend 64-bit builds and operating systems

**Q17. What is the NoSQL database? Is MongoDB a NoSQL database?**

**Ans.** From the advent of social media, the internet is being loaded with big data, big users, big complexity etc. and also becoming more complex day by day. NoSQL is the answer to all these problems; it is not a traditional database management system, not even a[**relational database management system (RDBMS)**](https://data-flair.training/blogs/sql-rdbms-database/). NoSQL stands for “Not Only SQL”. NoSQL is a kind of database that can handle and segregate all types of messy, unstructured, and complicated data. It is just a new way to describe the database. Yes, MongoDB is a NoSQL database.

**Q18. How to find values in the array with multiple criteria?**

**Ans.** Suppose you have the following document:

1. {\_id : 1, numbers : [-1000, 1000]}
2. {\_id : 2, numbers : [5]}

After this, if you write the following command :

1. db.example.**find**( { numbers : { $elemMatch : { $gt : -10 , $lt : 10 } } } );

It will return the \_id: 2 document.

**Q19. How to use select \* group by MongoDB aggregation?**

**Ans.** Suppose you want to select all attributes and groups by name across records. Let’s take an example:

1. { Name: George, x: 5, y: 3 }
2. { Name: George, z: 9 }
3. { Name: Rob, x: 12, y: 2 }

And now do the [**MongoDB aggregation**](https://data-flair.training/blogs/mongodb-aggregation-tutorial/) as follows:

1. db.example.**aggregate**(
2. { $group : {
3. \_id : “$Name”,
4. x: { $addToSet: “$x” },
5. y: { $addToSet: “$y” },
6. z: { $addToSet: “$z” },
7. }}
8. )

**Q20. How is sorting performed and what is a projection in MongoDB?**

**Ans.** Whenever we have to find any data from MongoDB, find() method is being used. This method will return the documents of the collection on which this is invoked. Now we can have where clause in MongoDB query so that we can [**limit the output by MongoDB projection**](https://data-flair.training/blogs/mongodb-projection/). Whenever we will execute the find() method MongoDB will return all documents related to that specific collection. Below is the syntax for same:

1. db.<collection\_name>.**find**({},{<Key\_Name>:<Flag to display>})

Here flag will have two values as 0 or 1. Here 1 indicates that the corresponding values will be displayed and 0 indicates that the corresponding values will not be displayed.

**Q21. Is it necessary to call “getLastError” to make a write durable?**

**Ans.** No, it is not necessary to call “getLastError”(Safe Mode). Here the server behaves as if it has been called. The “getLastError” simply allows one to get confirmation that write operation is successfully committed. Here you will want the confirmation, but the safety of write and its durability is independent.

**Q22. For how long does the replica set failover take?**

**Ans.** It takes around 10 – 30 seconds for the primary to be dropped down by the other members and a new primary to be elected. During this time span, the cluster is down for primary operations i.e writes and reads. Eventually, consistent queries can execute to secondaries at any time (in slaveOk mode), including during this window.

**Q23. What does the “\_v” field correspond to in Mongoose?**

**Ans.** When the mongoose is created for the first time the version key is a property set on each and every document. The value of this key has the internal revision of the document. It is understood that the name of this document is configurable. The default key is \_\_v

**Q24. Tell about composition/structure of ObjectId in MongoDB? Explain each composing element.**

[**ObjectId in MongoDB**](https://data-flair.training/blogs/mongodb-objectid-and-atomic-operations/) is very special. It associates with \_id field, and MongoDB uses objectid as the default value of \_id in documents. To generate objectid(), the following is the syntax:

1. **ObjectId**([SomeHexaDecimalValue])

Let’s take an example to understand it better:

1. obj\_id = new **ObjectId**()

The above query will give us the following output:  
ObjectId(“6549h6hgt8721s97u45798j7”)

There are some methods and details for the same are as follows:

* **str** – It provides a string representation of object id.
* **valueOf()** – It returns the hexadecimal representation of ObjectId.
* **getTimeStamp()** – It returns the creation timestamp of objected.
* **toString()** – It returns the string representation of ObjectId in the “ObjectId(haxstring)” .

**Q25. Tell about the capped collection. How we create them in MongoDB?**

**Ans.**[**Capped collection is a special type of collection in MongoDB**](https://data-flair.training/blogs/mongodb-capped-collection/). That means in this collection we can restrict the size of collection or we can put the limit on the size of the collection, these special type of collection is named as the capped collection. Here is the syntax for creating collections:

1. db.**createCollection**(<Collection\_Name>, {capped: Boolean, autoIndexId: Boolean, size: Number, max : Number})

Let’s understand the syntax.

* **Collection\_Name** – Name of the collection that we create as the capped collection.
* **capped** – Capped is the Boolean flag, true if the collection is to create as the capped collection, by default the value is false.
* **autoIndexId** – The Boolean flag that we use for auto indexing. If true indexes will create automatically, false will say indexes will not create automatically.
* **size** – Size parameter depicts the maximum size of documents in bytes. It is a mandatory field in terms of capped collections.
* **max** – max parameter depicts the maximum number of documents that allow in a collection. Size gets preference over size.

Here is the syntax for isCapped() method:

1. db.<Collection\_Name>.**isCapped**()

**Q 26. How can MongoDB simulate join or subquery?**

**Ans.** We are trying to figure out the best way to structure data in Mongo to simulate what would be a simple join or subquery in SQL.

Say we have the classic Users and Posts example, with Users in one collection and Posts in another. We want to find all posts by users who’s city is “Mumbai”.

We have simplified things in this question, in real-world scenario storing posts as an array in the user document won’t work as we have 1,000’s of “posts” per user constantly inserting.

**Q27. How can you sort a user-defined function; e.g. suppose x and y are integers, and the difference is (x-y)?**

**Ans.** It is not possible directly.  
Server-side sort:

1. db.**eval**(**function**() {
2. return db.scratch.**find**().**toArray**().**sort**(**function**(doc1, doc2) {
3. return doc1.a – doc2.a
4. })
5. });
6. Versus the equivalent client-side sort:
7. db.scratch.**find**().**toArray**().**sort**(**function**(doc1, doc2) {
8. return doc1.a – doc2.b
9. });

Note that it’s also possible to sort via an aggregation pipeline and by the $orderby operator (i.e. in addition to .sort()) however neither of these ways lets you provide a custom sort function either.

**Q28. How to update \_id of one MongoDB Document?**

**Ans.** If you want to update \_id then you will get something like this:

1. db.clients.**update**({‘\_id’:**ObjectId**(“4cc45467c55f4d2d2a000002”)}, {‘$set’:{‘\_id’:**ObjectId**(“4c8a331bda76c559ef000004”)}});

Mod on \_id not allowed

And the [**update of the document**](https://data-flair.training/blogs/mongodb-update-document/)is not made. You cannot update it by the given code. You’ll have to save the document using a new \_id, and then remove the old document.

Skip code block

1. // store the document in a variable
2. doc = db.clients.**findOne**({\_id: **ObjectId**(“4cc45467c55f4d2d2a000002”)})
3. // set a new \_id on the document
4. doc.\_id = **ObjectId**(“4c8a331bda76c559ef000004”)
5. // insert the document, using the new \_id
6. db.clients.**insert**(doc)
7. // remove the document with the old \_id
8. db.clients.**remove**({\_id: **ObjectId**(“4cc45467c55f4d2d2a000002”)})

To do it for your whole collection you can also use a loop

1. db.status.**find**().**forEach**(**function**(doc){ var id=doc.\_id; doc.\_id=doc.UserId; db.status.**insert**(doc); db.status.**remove**({\_id:id}); })

In this case, UserId was the new ID I wanted to use

**Q29.  How to get MongoDB database in JavaScript array?**

**Ans.** To get this use the following code:

1. use admin
2. dbs = db.**runCommand**({listDatabases: 1})
3. dbNames = []
4. **for** (var i in dbs.databases) { dbNames.**push**(dbs.databases[i].name) }
5. Hopefully this will help.
6. The below will create an array of the names of the database:
7. var connection = new **Mongo**();
8. var dbNames = connection.**getDBNames**();

**Q30. How to create a database in MongoDB?**

**Ans.** When I was trying to [**create a database in MongoDB**](https://data-flair.training/blogs/mongodb-create-database/), I got one error:

:~$ mongo  
MongoDB shell version: 1.6.5  
connecting to: test  
Error: couldn’t connect to server 127.0.0.1 (anon):1154  
exception: connect failed

A solution to the above issue:

1. cd /var/lib/mongodb/
2. Remove mongod.lock file from this folder
3. Sudo start mongodb (in console)
4. Mongo (in console)

And it runs fine.

First, you’ll need to run mongod on one terminal. Then fire up another terminal and type mongo. This shall open the mongo shell. You also need to create /data/db/ where mongo will store your databases.

This comes to the end of MongoDB interview questions and answers. We hope you liked our explanation.

What are NoSQL databases? What are the different types of NoSQL databases?

A NoSQL database provides a mechanism for storage and retrieval of data that is modeled in means other than the tabular relations used in relational databases (like SQL, Oracle, etc.).

Types of NoSQL databases:

* Document Oriented
* Key Value
* Graph
* Column Oriented

What kind of NoSQL database MongoDB is?

MongoDB is a document oriented database. It stores data in the form of BSON structure based documents. These documents are stored in a collection.

Which are the most important features of MongoDB?

* Flexible data model in form of documents
* Agile and highly scalable database
* Faster than traditional databases
* Expressive query language

What is a Namespace in MongoDB?

A Namespace is the concatenation of the database name and collection name. For e.g. school.students with school as the database and students as the collection

Which all languages can be used with MongoDB?

Currently, MonggoDB provides official driver support for C, C++, C#, Java, Node.js, Perl, PHP, Python, Ruby, Scala, Go and Erlang. MongoDB can easily be used with any of these languages. There are some other community supported drivers too but the above mentioned ones are officially provided by MongoDB.

Compare SQL databases and MongoDB at a high level.

SQL databases store data in form of tables, rows, columns and records. This data is stored in a pre-defined data model which is not very much flexible for today's real-world highly growing applications. MongoDB in contrast uses a flexible structure which can be easily modified and extended.

How is MongoDB better than other SQL databases?

MongoDB allows a highly flexible and scalable document structure. For e.g. one data document in MongoDB can have five columns and the other one in the same collection can have ten columns. Also, MongoDB database are faster as compared to SQL databases due to efficient indexing and storage techniques.

Compare MongoDB and CouchDB at high level.

Although both of these databases are document oriented, MongoDB is a better choice for applications which need dynamic queries and good performance on a very big database. On the other side, CouchDB is better used for applications with occasionally changing queries and pre-defined queries.

Does MongoDB support foreign key constraints?

No. MongoDB does not support such relationships.

Does MongoDB support ACID transaction management and locking functionalities?

No. MongoDB does not support default multi-document ACID transactions. However, MongoDB provides atomic operation on a single document.

How can you achieve primary key - foreign key relationships in MongoDB?

By default MongoDB does not support such primary key - foreign key relationships. However, we can achieve this concept by embedding one document inside another. Foe e.g. an address document can be embedded inside customer document.

Does MongoDB need a lot of RAM?

No. MongoDB can be run even on a small amount of RAM. MongoDB dynamically allocates and de-allocates RAM based on the requirements of other processes.

Does MongoDB pushes the writes to disk immediately or lazily?

MongoDB pushes the data to disk lazily. It updates the immediately written to the journal but writing the data from journal to disk happens lazily.

Explain the structure of ObjectID in MongoDB.

ObjectID is a 12-byte BSON type with:

* 4 bytes value representing seconds
* 3 byte machine identifier
* 2 byte process id
* 3 byte counter

MongoDB uses BSON to represent document structures. True or False?

True

If you remove a document from database, does MongoDB remove it from disk?

Yes. Removing a document from database removes it from disk too.

Mention the command to insert a document in a database called school and collection called persons.

use school;

db.persons.insert( { name: "kadhir", dept: "CSE" } )

What are Indexes in MongoDB?

Indexes support the efficient execution of queries in MongoDB. Without indexes, MongoDB must perform a collection scan, i.e. scan every document in a collection, to select those documents that match the query statement. If an appropriate index exists for a query, MongoDB can use the index to limit the number of documents it must inspect.

How many indexes does MongoDB create by default for a new collection?

By default, MongoDB created the \_id collection for every collection.

Can you create an index on an array field in MongoDB? If yes, what happens in this case?

Yes. An array field can be indexed in MongoDB. In this case, MongoDB would index each value of the array.

What is a covered query in MongoDB?

A covered query is the one in which:

* fields used in the query are part of an index used in the query, and
* the fields returned in the results are in the same index

Why is a covered query important?

Since all the fields are covered in the index itself, MongoDB can match the query condition as well as return the result fields using the same index without looking inside the documents. Since indexes are stored in RAM or sequentially located on disk, such access is a lot faster.

Does MongoDB provide a facility to do text searches? How?

Yes. MongoDB supports creating text indexes to support text search inside string content. This was a new feature which can introduced in version 2.6.

What happens if an index does not fit into RAM?

If the indexes do not fit into RAM, MongoDB reads data from disk which is relatively very much slower than reading from RAM.

Mention the command to list all the indexes on a particular collection.

db.collection.getIndexes()

At what interval does MongoDB write updates to the disk?

By default configuration, MongoDB writes updates to the disk every 60 seconds. However, this is configurable with the commitIntervalMs and syncPeriodSecs options.

How can you achieve transaction and locking in MongoDB?

To achieve concepts of transaction and locking in MongoDB, we can use the nesting of documents, also called embedded documents. MongoDB supports atomic operations within a single document.

What is Aggregation in MongoDB?

Aggregations 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 and commands.

What is Sharding in MongoDB? Explain.

Sharding is a method for storing data across multiple machines. MongoDB uses sharding to support deployments with very large data sets and high throughput operations.

What is Replication in MongoDB? Explain.

Replication is the process of synchronizing data across multiple servers. Replication provides redundancy and increases data availability. With multiple copies of data on different database servers, replication protects a database from the loss of a single server. Replication also allows you to recover from hardware failure and service interruptions.

What are Primary and Secondary Replica sets?

Primary and master nodes are the nodes that can accept writes. MongoDB's replication is 'single-master:' only one node can accept write operations at a time.

Secondary and slave nodes are read-only nodes that replicate from the primary.

By default, MongoDB writes and reads data from both primary and secondary replica sets. True or False.

False. MongoDB writes data only to the primary replica set.

Why are MongoDB data files large in size?

MongoDB preallocates data files to reserve space and avoid file system fragmentation when you setup the server.

When should we embed one document within another in MongoDB?

You should consider embedding documents for:

* 'contains' relationships between entities
* One-to-many relationships
* Performance reasons

Why MongoDB is not preferred over a 32-bit system?

When running a 32-bit build of MongoDB, the total storage size for the server, including data and indexes, is 2 gigabytes. For this reason, do not deploy MongoDB to production on 32-bit machines.

If you're running a 64-bit build of MongoDB, there's virtually no limit to storage size.

What is a Storage Engine in MongoDB

A storage engine is the part of a database that is responsible for managing how data is stored on disk. For example, one storage engine might offer better performance for read-heavy workloads, and another might support a higher-throughput for write operations.

Which are the two storage engines used by MongoDB?

MongoDB uses MMAPv1 and WiredTiger.

What is the role of a profiler in MongoDB? Where does the writes all the data?

The database profiler collects fine grained data about MongoDB write operations, cursors, database commands on a running mongod instance. You can enable profiling on a per-database or per-instance basis.

The database profiler writes all the data it collects to the system.profile collection, which is a capped collection.

How does Journaling work in MongoDB?

When running with journaling, MongoDB stores and applies write operations in memory and in the on-disk journal before the changes are present in the data files on disk. Writes to the journal are atomic, ensuring the consistency of the on-disk journal files. With journaling enabled, MongoDB creates a journal subdirectory within the directory defined by dbPath, which is /data/db by default.

Mention the command to check whether you are on the master server or not.

db.isMaster()

Can you configure the cache size for MMAPv1? How?

No. MMAPv1 does not allow configuring the cache size.

Can you configure the cache size for WiredTiger? How?

For the WiredTiger storage engine, you can specify the maximum size of the cache that WiredTiger will use for all data. This can be done using storage.wiredTiger.engineConfig.cacheSizeGB option.

How does MongoDB provide concurrency?

MongoDB uses reader-writer locks that allow concurrent readers shared access to a resource, such as a database or collection, but give exclusive access to a single write operation.

How can you isolate your cursors from intervening with the write operations?

You can use the snapshot() method on a cursor to isolate the operation for a very specific case. snapshot() traverses the index on the \_id field and guarantees that the query will return each document no more than once.

Can one MongoDB operation lock more than one databases? If yes, how?

Yes. Operations like copyDatabase(), repairDatabase(), etc. can lock more than onne databases involved.

How can concurrency affect replica sets primary?

In replication, when MongoDB writes to a collection on the primary, MongoDB also writes to the primary's oplog, which is a special collection in the local database. Therefore, MongoDB must lock both the collection's database and the local database.

What is GridFS?

GridFS is a specification for storing and retrieving files that exceed the BSON-document size limit of 16MB. Instead of storing a file in a single document, GridFS divides a file into parts, or chunks, and stores each of those chunks as a separate document.

Can you run multiple Javascript operations in a single mongod instance?

Yes. The V8 JavaScript engine added in 2.4 allows multiple JavaScript operations to run at the same time.

Which command can be used to provide various information on the query plans used by a MongoDB query?

The explain() command can be used for this information. The possible modes are: 'queryPlanner', 'executionStats', and 'allPlansExecution'.

MongoDB Interview Questions

1) What do you understand by NoSQL databases? Is MongoDB a NoSQL database? explain.

At the present time, the internet is loaded with big data, big users, big complexity etc. and also becoming more complex day by day. NoSQL is answer of all these problems, It is not a traditional database management system, not even a relational database management system (RDBMS). NoSQL stands for "Not Only SQL". NoSQL is a type of database that can handle and sort all type of unstructured, messy and complicated data. It is just a new way to think about the database.

Yes. MongoDB is a NoSQL database.

2) Which are the different languages supported by MongoDB?

MonggoDB provides official driver support for C, C++, C#, Java, Node.js, Perl, PHP, Python, Ruby, Scala, Go and Erlang.

You can use MongoDB with any of the above languages. There are some other community supported drivers too but the above mentioned ones are officially provided by MongoDB.

3) What are the different types of NoSQL databases? Give some example.

NoSQL database can be classified as 4 basic types:

1. Key value store NoSQL database
2. Document store NoSQL database
3. Column store NoSQL database
4. Graph base NoSQL databse

There are many NoSQL databases. MongoDB, Cassandra, CouchBD, Hypertable, Redis, Riak, Neo4j, HBASE, Couchbase, MemcacheDB, Voldemort, RevenDB etc. are the examples of NoSQL databases.

4) Is MongoDB better than other SQL databases? If yes then how?

MongoDB is better than other SQL databases because it allows a highly flexible and scalable document structure.

**For example:**

* One data document in MongoDB can have five columns and the other one in the same collection can have ten columns.
* MongoDB database are faster than SQL databases due to efficient indexing and storage techniques.

5) What type of DBMS is MongoDB?

MongoDB is a document oriented DBMS

6) What is the difference between MongoDB and MySQL?

Although MongoDB and MySQL both are free and open source databases, there is a lot of difference between them in the term of data representation, relationship, transaction, querying data, schema design and definition, performance speed, normalization and many more. To compare MySQL with MongoDB is like a comparison between Relational and Non-relational databases.

7) Why MongoDB is known as best NoSQL database?

MongoDb is the best NoSQL database because, it is:

Document Oriented

Rich Query language

High Performance

Highly Available

Easily Scalable

8) Does MongoDB support primary-key, foreign-key relationship?

No. By Default, MongoDB doesn't support primary key-foreign key relationship.

9) Can you achieve primary key - foreign key relationships in MongoDB?

We can achieve primary key-foreign key relationship by embedding one document inside another. For example: An address document can be embedded inside customer document.

10) Does MongoDB need a lot of RAM?

No. There is no need a lot of RAM to run MongoDB. It can be run even on a small amount of RAM because it dynamically allocates and de-allocates RAM according to the requirement of the processes.

11) Explain the structure of ObjectID in MongoDB.

ObjectID is a 12-byte BSON type. These are:

* 4 bytes value representing seconds
* 3 byte machine identifier
* 2 byte process id
* 3 byte counter

12) Is it true that MongoDB uses BSON to represent document structure?

Yes.

13) What are Indexes in MongoDB?

In MondoDB, Indexes are used to execute query efficiently. Without indexes, MongoDB must perform a collection scan, i.e. scan every document in a collection, to select those documents that match the query statement. If an appropriate index exists for a query, MongoDB can use the index to limit the number of documents it must inspect.

14) By default, which index is created by MongoDB for every collection?

By default, the\_id collection is created for every collection by MongoDB.

15) What is a Namespace in MongoDB?

Namespace is a concatenation of the database name and the collection name. Collection, in which MongoDB stores BSON objects.

16) Can journaling features be used to perform safe hot backups?

Yes.

17) Why does Profiler use in MongoDB?

MongoDB uses a database profiler to perform characteristics of each operation against the database. You can use a profiler to find queries and write operations

18) If you remove an object attribute, is it deleted from the database?

Yes, it be. Remove the attribute and then re-save() the object.

19) In which language MongoDB is written?

MongoDB is written and implemented in C++.

20) Does MongoDB need a lot space of Random Access Memory (RAM)?

No. MongoDB can be run on small free space of RAM.

21) What language you can use with MongoDB?

MongoDB client drivers supports all the popular programming languages so there is no issue of language, you can use any language that you want.

22) Does MongoDB database have tables for storing records?

No. Instead of tables, MongoDB uses "Collections" to store data.

23) Do the MongoDB databases have schema?

Yes. MongoDB databases have dynamic schema. There is no need to define the structure to create collections.

24) What is the method to configure the cache size in MongoDB?

MongoDB's cache is not configurable. Actually MongoDb uses all the free spaces on the system automatically by way of memory mapped files.

25) How to do Transaction/locking in MongoDB?

MongoDB doesn't use traditional locking or complex transaction with Rollback. MongoDB is designed to be light weighted, fast and predictable to its performance. It keeps transaction support simple to enhance performance.

26) Why 32 bit version of MongoDB are not preferred ?

Because MongoDB uses memory mapped files so when you run a 32-bit build of MongoDB, the total storage size of server is 2 GB. But when you run a 64-bit build of MongoDB, this provides virtually unlimited storage size. So 64-bit is preferred over 32-bit.

27) Is it possible to remove old files in the moveChunk directory?

Yes, These files can be deleted once the operations are done because these files are made as backups during normal shard balancing operation. This is a manual cleanup process and necessary to free up space.

28) What will have to do if a shard is down or slow and you do a query?

If a shard is down and you even do query then your query will be returned with an error unless you set a partial query option. But if a shard is slow them Mongos will wait for them till response.

29)Explain the covered query in MongoDB.

A query is called covered query if satisfies the following two conditions:

* The fields used in the query are part of an index used in the query.
* The fields returned in the results are in the same index.

30) What is the importance of covered query?

Covered query makes the execution of the query faster because indexes are stored in RAM or sequentially located on disk. It makes the execution of the query faster.

Covered query makes the fields are covered in the index itself, MongoDB can match the query condition as well as return the result fields using the same index without looking inside the documents.

31) What is sharding in MongoDB?

In MongoDB, Sharding is a procedure of storing data records across multiple machines. It is a MongoDB approach to meet the demands of data growth. It creates horizontal partition of data in a database or search engine. Each partition is referred as shard or database shard.

32) What is replica set in MongoDB?

A replica can be specified as a group of mongo instances that host the same data set. In a replica set, one node is primary, and another is secondary. All data is replicated from primary to secondary nodes.

33) What is primary and secondary replica set in MongoDB?

In MongoDB, primary nodes are the node that can accept write. These are also known as master nodes. The replication in MongoDB is single master so, only one node can accept write operations at a time.

Secondary nodes are known as slave nodes. These are read only nodes that replicate from the primary.

34) By default, which replica sets are used to write data?

By default, MongoDB writes data only to the primary replica set.

35) What is CRUD in MongoDB?

MongoDB supports following CRUD operations:

* Create
* Read
* Update
* Delete

36) In which format MongoDB represents document structure?

MongoDB uses BSON to represent document structures.

37) What will happen when you remove a document from database in MongoDB? Does MongoDB remove it from disk?

Yes. If you remove a document from database, MongoDB will remove it from disk too.

38) Why are MongoDB data files large in size?

MongoDB doesn't follow file system fragmentation and pre allocates data files to reserve space while setting up the server. That's why MongoDB data files are large in size.

39) What is a storage engine in MongoDB?

A storage engine is the part of a database that is used to manage how data is stored on disk.

**For example:** one storage engine might offer better performance for read-heavy workloads, and another might support a higher-throughput for write operations.

40) Which are the storage engines used by MongoDB?

MMAPv1 and WiredTiger are two storage engine used by MongoDB.

41) What is the usage of profiler in MongoDB?

A database profiler is used to collect data about MongoDB write operations, cursors, database commands on a running mongod instance. You can enable profiling on a per-database or per-instance basis.

The database profiler writes all the data it collects to the system. profile collection, which is a capped collection.

42) Is it possible to configure the cache size for MMAPv1 in MongoDB?

No. it is not possible to configure the cache size for MMAPv1 because MMAPv1 does not allow configuring the cache size.

43) How to configure the cache size for WiredTiger in MongoDB?

For the WiredTiger storage engine, you can specify the maximum size of the cache that WiredTiger will use for all data. This can be done using storage.wiredTiger.engineConfig.cacheSizeGB option.

44) How does MongoDB provide concurrency?

MongoDB uses reader-writer locks for concurrency. Reader-writer locks allow concurrent readers shared access to a resource, such as a database or collection, but give exclusive access to a single write operation.

45) What is the difference between MongoDB and Redis database?

**Difference between MongoDB and Redis:**

* Redis is faster than MongoDB.
* Redis has a key-value storage whereas MongoDB has a document type storage.
* Redis is hard to code but MongoDB is easy.

For more information: [click here](http://www.javatpoint.com/redis-vs-mongodb)

46) What is the difference between MongoDB and CouchDB?

**Difference between MongoDB and CouchDB:**

* MongoDB is faster than CouchDB while CouchDB is safer than MongoDB.
* Triggers are not available in MongoDB while triggers are available in CouchDB.
* MongoDB serializes JSON data to BSON while CouchDB doesn't store data in JSON format.

For more information: [click here](http://www.javatpoint.com/couchdb-vs-mongodb)

47) What is the difference between MongoDB and Cassandra?

**Difference between MongoDB and Cassandra:**

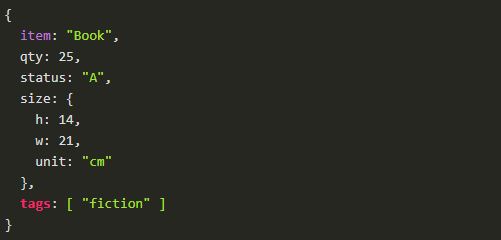
* MongoDB is cross-platform document-oriented database system while Cassandra is high performance distributed database system.
* MongoDB is written in C++ while Cassandra is written in Java.
* MongoDB is easy to administer in the case of failure while Cassandra provides high availability with no single point of failure.

**Q1. What is MongoDB and Why we need MongoDB?**

MongoDB is the most famous NoSQL open source database management system. It is written in C++ language and developed by MongoDB Inc.

MongoDB is a document oriented database which means it stores the data in BSON format which is a binary representation of JSON and it contains more data types than JSON.

https://miro.medium.com/max/60/1*dCL7745rtKdeZkwYHSz7IQ.png?q=20



In MongoDB, tables are called as collection, rows are called as document, columns are called as field, joins are called as linking.

MongoDB is widely supported by most of the languages like, JavaScript, Java, C# and runs on all the available OS.

**Q2. What is difference between MongoDB and SQL or NoSQL between SQL?**

In SQL, data is stored in Table format whereas in MongoDB, it stores in JSON format.

In MongoDB, development is simplified as its documents map naturally to modern, object-oriented programming languages. In SQL, we need object-relational mapping (ORM) layer that translates objects in code to relational tables.

The MongoDB provides high performance, high availability, easy scalability rather than SQL Server.

In MongoDB, each document can store data with different attributes from other documents. With JSON documents, we can add new attributes when we need to, without having to alter a centralized database schema. But in a relational database, this causes downtime and significant performance overhead

Having all the data for an object in one place also makes it easier for developers to understand and optimize query performance.

When using JSON data, MySQL drivers do not have the capability to properly and precisely convert JSON into a useful native data type used by the application. This includes different types of numeric values (e.g. floating points, 64-bit integers, decimals) timestamps, and dates, or a Map or List in Java or a Dictionary or List in Python. Instead developers have to manually convert text-based JSON in their application, losing the ability to have fields that can take on multiple data types in different documents (polymorphism) and making the computation, sorting and comparison of values difficult and error-prone. But MongoDB and its drivers supports advanced data types not supported by regular text-based JSON.

**Q3. What is BSON in MongoDB?**

MongoDB stores data as BSON documents. BSON is a binary representation of JSON documents, though it contains more data types than JSON. Some of the supported data types are: Double, String, Object, Array, Binary Data, RegEx, Boolean, Date, Null, JavaScript, Timestamp. ObjectId.

Some of the features of BSON:

* BSON is lightweight and is an important feature for any data representation format, especially when used over the network.
* BSON is designed to be traversed easily.
* It is efficient. Encoding data to BSON and decoding from BSON can be performed very quickly in most languages

**Q4. What is \_id Field in MongoDB?**

In MongoDB, each document stored in a collection requires a unique \_idfield that acts as a primary key. If an inserted document omits the \_id field, the MongoDB driver automatically generates an ObjectId for the \_id field.

The \_id field has the following behavior and constraints:

* By default, MongoDB creates a unique index on the \_id field during the creation of a collection.
* The \_id field is always the first field in the documents. If the server receives a document that does not have the \_id field first, then the server will move the field to the beginning.
* The \_id field may contain values of any BSON data type, other than an array.

While storing values for \_id, we can follow these options: Use an ObjectId, Use a natural unique identifier, Generate an auto-incrementing number, Generate a UUID in our application code.

Most MongoDB driver clients will include the \_id field and generate an ObjectId before sending the insert operation to MongoDB; however, if the client sends a document without an \_id field, the mongod will add the \_idfield and generate the ObjectId.

**Q5. How do you create or select a database in MongoDB?**

In MongoDB, databases hold collections of documents. To select a database to use issue the use <db> statement, as in the following example:

https://miro.medium.com/max/60/1*-bWvI51_leERSnaJBBFDtA.png?q=20

https://miro.medium.com/max/290/1*-bWvI51_leERSnaJBBFDtA.png

If a database does not exist, MongoDB creates the database when we first store data for that database. Some points to be noted when creating database in MongoDB are:

* Database Name Case Sensitivity
* For MongoDB deployments running on Windows, database names cannot contain any of the following characters: /\. "$\*<>:|?
* For MongoDB deployments running on Unix and Linux systems, database names cannot contain any of the following characters: /\. "$
* Database names cannot be empty and must have fewer than 64 characters.

**Q6. What is collection in MongoDB?**

A *collection* is a group of documents. If a document is the MongoDB analog of a row in a relational database, then collection is the analog to a table.

Collections have *dynamic schemas*. This means that the documents within a single collection can have any number of different shapes. For example, both the following documents can be stored in a single collection.

https://miro.medium.com/max/60/1*_tVsw3UEaqFzC3UWJCoIcA.png?q=20

https://miro.medium.com/max/459/1*_tVsw3UEaqFzC3UWJCoIcA.png

Note that the previous documents not only have different types for their values (string versus integer) but also have entirely different keys. Because in MongoDB any document can be put into any collection.

**Q7. What is writeConcern?**

Whenever we query the database to insert(), insertOne() or insertMany(), we can pass an additional parameter which is writeConcern. It is a document describes the level of acknowledgement requested from MongoDB for write operations. Write concern can include the following fields:

https://miro.medium.com/max/60/1*kMCmhbrTx9C7vTLpBr3ozw.png?q=20

https://miro.medium.com/max/315/1*kMCmhbrTx9C7vTLpBr3ozw.png

* the w option to request acknowledgement that the write operation has propagated to a specified number of mongod instances or to mongodinstances with specified tags.
* the j option to request acknowledgement that the write operation has been written to the journal, and
* the wtimeout option to specify a time limit to prevent write operations from blocking indefinitely.

**Q8. What is upsert?**

upsert is one of the option that we can pass to the update operation as third parameter.

If upsert is true and no document matches the query criteria, update()inserts a *single* document. The update creates the new document with either:

* The fields and values of the update parameter if the update parameter is a replacement document (i.e., contains only field and value pairs). If neither the query nor the update document specifies an \_id field, MongoDB adds the \_id field with an ObjectId value.
* The fields and values of both the query and update parameters if the update parameter contains update operator expressions (such as $inc, $set, $min, $max). The update creates a base document from the equality clauses in the parameter, and then applies the update expressions from the updateparameter. Comparison operations from the query will not be included in the new document.

Also if we want to update multiple documents we can pass multi: true as another option.

**Q9. Explain some of the Evaluation Query Operators.**

* **$expr**: $expr can build query expressions that compare fields from the same document in a $match stage. Example: db.users.find( { $expr: { $gt: [ "$age", "$requiredAge" ] } } ). This query will finds documents where the age is greater than the requiredAge.
* **$jsonSchema**: $jsonSchema can be used in a document validator, which enforces that inserted or updated documents are valid against the schema.
* **$mod**: Select documents where the value of a field divided by a divisor has the specified remainder. It is performing a modulo operation. Example: db.users.find( { items: { $mod: [ 4, 0 ] } } )
* **$regex**: Provides regular expression capabilities for pattern matching strings in queries. MongoDB uses Perl compatible regular expressions. Example: db.users.find( { name: { $regex: /^John/ } } )
* **$text**: $text performs a text search on the content of the fields indexed with a text index. Example: db.users.find( { $text: { $search: "john" } } )
* **$where**: Use the $where operator to pass either a string containing a JavaScript expression or a full JavaScript function to the query system.

**Q10. What is Storage Engine? What are all the Storage Engines available in MongoDB?**

The storage engine is the component of the database that is responsible for managing how data is stored, both in memory and on disk. MongoDB supports multiple storage engines, as different engines perform better for specific workloads.

* **WiredTiger Storage Engine**: This is the default engine. It is well-suited for most workloads and is recommended for new deployments. WiredTiger provides a document-level concurrency model, checkpointing, and compression, among other features.
* **In-Memory Storage Engine**: In-Memory Storage Engine is available in MongoDB Enterprise. Rather than storing documents on-disk, it retains them in-memory for more predictable data latencies.
* **MMAPv1 Storage Engine**: This is deprecated as of MongoDB v4.0 and is the default storage engine for MongoDB versions 3.0 and earlier.

**1) What is Redis?**

Redis is an advanced key-value data store and cache. It has is also referred to as a data structure server as such the keys not only contains strings, but also hashes, sets, lists, and sorted sets. Companies using Redis includes StackOverflow, Twitter, Github, etc.

**2) Explain the Replication feature of Redis?**

Redis supports simple master to slave replication.  When a relationship is established, data from the master is transferred to the slave. Once this is done, all changes to the master replicate to the slave

**3) What is the difference between Memcached and Redis?**

|  |  |
| --- | --- |
| Redis | Memcached |
| * Redis also does cache information but has got additional features like persistence and replication * Redis does not support the functionality of LRU (least recently used) eviction of values * In Redis you can set a time out on everything when memory is full, it will look at three random keys and deletes the one which is closest to expiry * Redis does not support CAS ( Check and Set). It is useful for maintaining cache consistency * Redis has got stronger data structures; it can handle strings, binary safe strings, list of binary safe strings, sorted lists, etc. * Redis had a maximum of 2GB key length * Redis is single threaded | * Memcached only cache information. * Memcached supports the functionality of LRU (least recently used) eviction of values * In Memcached when they overflow memory, the one you have not used recently (LRU- least recently used) will get deleted * Memcached supports CAS (Check and Set) * In Memcached, you have to serialize the objects or arrays in order to save them and to read them back you have to un-serialize them. * Memcached had a maximum of 250 bytes length * Memcached is a multi-threaded |

**4) What are the advantages of using Redis?**

Advantage of using Redis are

* It provides high speed
* It supports a server-side locking
* It has got lots of client lib
* It has got command level Atomic Operation (tx operation)

**5) What are the limitations of Redis?**

* It is single threaded
* It has got limited client support for consistent hashing
* It has significant overhead for persistence
* It is not deployed widely

[](https://career.guru99.com/wp-content/uploads/2014/10/467px-Redis_Logo.svg_.png)

**6) List out the operation keys of Redis?**

Operation keys of Redis include

* TYPE key
* TTL key
* KEYS pattern
* EXPIRE key seconds
* EXPIREAT key timestamp
* EXISTS key
* DEL key

**7) Which PHP module can be used with Redis?**

In PHP module, PRedis is more preferable than Redid PHP binding or Resident

**8) Does Redis give speed and durability both?**

No, Redis purposely compromises the durability to enhance the speed. In Redis, in the event of system failure or crash, Redis writes to disk but may fall behind and lose the data which is not stored.

**9) How can you improve the durability in Redis?**

To improve the durability of Redis **“append only file”** can be configured by using fsync data on disk.

* Fsync () every time a new command is added to the append log file: It is safe but very slow
* Fysnc() one time every second: It is fast, but you may lose 1 second of data if system fails
* Never fsync(): It is an unsafe method, and your data is in hand of Operating System

**10) Mention what are the things you have to take care while using Redis?**

While using Redis one must take care of

* Select a consistent method to name and prefix your keys. Manage your namespace
* Create a “Registry” of key prefixes that maps each of your internal documents for that application which “own” them
* For every class you put through into your Redis infrastructure: design, implement and test the mechanisms for garbage collection or data migration to archival storage
* Design, implement and test a sharding library before you have invested much into your application deployment and make sure that you keep a registry of “shards “replicated on each server
* Separate all your K/V store and related operations into your own library/API or service

Redis Interview Questions

A list of frequently asked **Redis interview questions** and answers are given below.

1) What is Redis?

Redis is an open-source, advance key value data store and cache. It is also referred as a data structure server which keys not only contains strings, but also hashes, sets, lists, and sorted sets.

2) What is the meaning of Redis?

Redis stands for REmote DIctionary Server.

3) How is Redis different from other databases?

Redis is a NoSQL, Opensource, in-memory data-structure store. It follows the principle of key-value store.

It is extremely fast, persistent, portable and supports many languages such as C, C++, C#, Clojure, Common Lisp, D, Dart, Erlang, Go, Haskell, Haxe, Io, Java, JavaScript (Node.js), Julia, Lua, Objective-C, Perl, PHP, Pure Data, Python, R, Racket, Ruby, Rust, Scala, Smalltalk and Tcl.

**For more information:**[**Click Here**](https://www.javatpoint.com/redis-vs-other-key-value-stores)

4) In which language Redis is written?

Redis is written in ANSI C and mostly used for cache solution and session management. It creates unique keys for store values.

5) What is the usage of Redis?

Redis is a special key-value store database that can function as a NoSQL database or as a memory-cache store to improve performance when serving data that is stored in system memory.

6) How to interect with Redis?

After the installation of the server you can run the Redis Client provided by redis installation or you can open the command prompt and use the following command:

1. redis-cli

By using any of them, you can interect with Redis.

7) Which are the most popular companies using Redis?

Twitter, Github, Stackoverflow etc. are the most popular companies using Redis.

8) What are the main features of Redis?

Following are the main features of Redis:

* Redis is very simple to install setup and manage.
* Redis is very fast. It can execute 100000 queries per second.
* Redis is fast because data is being persistent in memory as well as stored on the disk.
* Redis is very fast because it loads the whole dataset in primary memory.
* Redis operations working on different data types are atomic so these operations can be accomplished safely i.e. to set or increase a key, add or remove elements from a set or increase a counter.
* It supports various types of data structure such as strings, hashes, sets, lists, sorted sets etc.
* Redis supports a variety of languages i.e. C, C++, C#, Ruby, Python, Twisted Python, PHP, Erlang, Tcl, Perl, Lua, Java, Scala etc.
* If your favorite language is not supported yet, you can write your own client library, as the Protocol is pretty simple.
* Redis supports simple master to slave replication.
* Redis is portable.

**For more information:**[**Click Here**](https://www.javatpoint.com/features-of-redis)

9) Explain the Replication Features of Redis?

Replication is important in order to archive high level of availability in big data systems. The data needs to be replicated at n number of places. This follows the master-slave approach where the master copy is maintained by master-slave and replicated to n other nodes.

10) What are the advantages of using Redis?

Following is a list of some important advantages of Redis:

* Redis is very fast.
* It supports a server-side locking.
* It has a rich client side library.
* It is a good counter.
* It supports Atomic Operation.

11) What are the disadvantages/limitations of using Redis?

Following are the disadvantages/ limitations of Redis:

* It is single threaded.
* It has got limited client support for consistent hashing.
* It has significant overhead for persistence.
* It is not deployed widely.

12) What is the difference between Redis and RDBMS?

There are a lot of differences between Redis and RDBMS:

* Redis is a NoSQL database while RDBMS is an SQL database.
* Redis follows the key-value structure while RDBMS follows the table structure.
* Redis extremely fast while RDBMS is comparatively slow.
* Redis stores all the dataset in primary memory while RDBMS stores its dataset in secondary memory.
* Redis is generally used to store small and frequently used files while RDBMS is used to store big files.
* Redis provides only official support for Linux, BSD, Mac OS X, Solaris. It doesn?t provide official support for Windows currently while RDBMS provides support for both.

**For more information:**[**Click Here**](https://www.javatpoint.com/redis-vs-rdbms)

13) Differentiate Memcached and Redis?

|  |  |
| --- | --- |
| **Memcached** | **Redis** |
| Memcached only does caching information. | It provides some more functionalities like replication and persistence along with caching information. |
| Memcached supports the functionality of LRU (Least Recently Used) eviction of values. | LRU is not supported by Redis. |
| In Memcached, when they overflow memory, the one you have not used recently (LRU- Least Recently Used) will get deleted. | In Redis, there is a time set for each function, Three keys are maintained, the one, which is closest to expiry, will get deleted. |
| CAS (Check and Set) is supported by Memcached. | CAS is not supported by Redis. |
| Array objects are needed to be serialized in order to get saved. We need to unserialize them for their retrieval. | Redis has got stronger data structures; it can handle strings, binary safe strings, list of binary safe strings, sorted lists, etc. |
| Memcached has at most 250 bytes length. | Redis has at most 2 GB key length. |
| It is Multi-threaded | It is single threaded. |

14) Mention Operation Keys of Redis.

* TYPE key
* TTL key
* EXPIRE key seconds
* EXPIREAT key timestamp
* EXISTS key
* DEL key

15) Which are the different data types used in Redis?

There are mainly 5 types of data types supported by Redis:

* Strings
* Hashes
* Lists
* Sets
* Sorted Sets

**For more information:**[**Click Here**](https://www.javatpoint.com/redis-data-types)

16) Which are the most popular commands of the Redis database?

**For more information:**[**Click Here**](https://www.javatpoint.com/redis-all-commands)

17) We all know that Reds is fast, but is it also durable?

In Redis, there is always a trade-of between durability and speed. In the case of system failure, it may lose data which is not stored.

18) How can you enhance the durability of Redis?

* Whenever a new command is added to the append log file, call Fsysnc() each time.
* Keep calling Fsysnc() in every second. Despite of the 1 second data lose in the case of system fails.

19) What are the concerned things while using Redis?

* Consistent method selection in order to name and prefix the keys. Namespace management.
* Make key prefixes registry which can map every documents to their owner applications.
* Designing, implementing and testing the Garbage collection mechanism for each class we keep into the redis architecture.
* Maintain a sharding library before investing so much into the application.

20) How can you use Redis with .Net application?

To use Redis in .Net applications, follow these steps:

* First, Download Redis Server.
* Install Redis Server.
* Download Redis Client.
* Set Configuration into Web.config File.
* Use Redis Client Class.

1. **Question 1. What Is Redis?**

**Answer :**

Redis is an open source released under BSD licensed, in-memory data structure store, it can be used as a database, cache and message broker.

1. **Question 2. What Is Main Feature Of Redis?**

**Answer :**

* + Data is being persistent in-memory as well as stored on disk that’s make Redis fast.
  + Redis supports simple master to slave replication.
  + Redis can execute 1, 00,000 queries per second.

[Python Interview Questions](https://www.wisdomjobs.com/e-university/python-interview-questions.html)

1. **Question 3. What Is Advantages Of Using Redis?**

**Answer :**

* + High speed
  + Server-side locking
  + Rich Client Side Library
  + Support Atomic Operation

1. **Question 4. List The Data Structures Supported By Redis?**

**Answer :**

**Redis supports following Data Structures:**

* + Strings
  + Hashes
  + Lists
  + Sets
  + Sorted sets with range queries
  + Bitmaps
  + Hyperloglogs
  + Geospatial indexes with radius queries

[Python Tutorial](https://www.wisdomjobs.com/e-university/python-tutorial-176.html" \o "Python Tutorial)

1. **Question 5. What Is Limitations Of Redis?**

**Answer :**

* + Single threaded
  + It has got limited client support for consistent hashing
  + It has significant overhead for persistence
  + It is not deployed widely

[RDBMS Interview Questions](https://www.wisdomjobs.com/e-university/rdbms-interview-questions.html" \o "RDBMS Interview Questions)

1. **Question 6. List Out The Operation Keys Of Redis?**

**Answer :**

**Operation keys of Redis includes:**

* + TYPE key
  + TTL key
  + KEYS pattern
  + EXPIRE key seconds
  + EXPIREAT key timestamp
  + EXISTS key
  + DEL key

1. **Question 7. How We Can Use Redis With .net Application?**

**Answer :**

**For use Redis in .net application need to follow given steps:**

* + Download Redis Server
  + Install Redis Server
  + Download Redis Client
  + Set Configuration into Web.config File
  + Use Redis Client Class

[PHP Tutorial](https://www.wisdomjobs.com/e-university/php-tutorial-223.html" \o "PHP Tutorial) [PHP Interview Questions](https://www.wisdomjobs.com/e-university/php-interview-questions.html" \o "PHP Interview Questions)

1. **Question 8. What Is Redis-cli?**

**Answer :**

Redis-cli is the Redis command line interface, a simple program that allows sending commands to Redis, and read the replies sent by the server, directly from the terminal.

1. **Question 9. Explain Repl?**

**Answer :**

REPL stands for Read Eval Print Loop. It an interactive mode where the user types commands and get replies.

[MySQL Interview Questions](https://www.wisdomjobs.com/e-university/mysql-interview-questions.html" \o "MySQL Interview Questions)

1. **Question 10. What The Basic Features Are Of Redis Which Makes It Awesome Compared To Memcache?**

**Answer :**

* + Powerful data types and powerful commands to leverage them. Hashes, Sorted Sets, Lists, and more.
  + Persistence to disk, by default.
  + Transactions with optimistic locking (WATCH/MULTI/EXEC)
  + Pub/sub. Extremely fast.
  + Values up to 512MB in size (memcached limited to 1MB per key)
  + Lua scripting (as of 2.6)
  + Extremely fast at everything. Benchmarks are often conflicting, but this much is clear: when used like memcached Redis falls somewhere between nearly as fast or maybe even a little faster. Like memcached it is often bound by available network or even memory bandwidth instead of CPU or other bottlenecks and will rarely be the culprit when your app is slowing down.

[MySQL Tutorial](https://www.wisdomjobs.com/e-university/mysql-tutorial-275.html" \o "MySQL Tutorial)

1. **Question 11. How Can You Use Redis With .net Application?**

**Answer :**

**To use Redis in .Net applications, follow these steps:**

* + First, Download Redis Server.
  + Install Redis Server.
  + Download Redis Client.
  + Set Configuration into Web.config File.
  + Use Redis Client Class.

[PHP and Jquery Interview Questions](https://www.wisdomjobs.com/e-university/php-and-jquery-interview-questions.html" \o "PHP and Jquery Interview Questions)

1. **Question 12. We All Know That Reds Is Fast, But Is It Also Durable?**

**Answer :**

No. Redis compromises with durability to enhance the speed. In Redis, in the case of system failure or crash, it writes to disk but may fall behind and lose the data which is not stored.

[Python Interview Questions](https://www.wisdomjobs.com/e-university/python-practice-tests-176-326595" \o "Python Interview Questions)

1. **Question 13. What Are The Main Features Of Redis?**

**Answer :**

**Following are the main features of Redis:**

* + Redis is very simple to install setup and manage.
  + Redis is very fast. It can execute 100000 queries per second.
  + Redis is fast because data is being persistent in memory as well as stored on the disk.
  + Redis operations working on different data types are atomic so these operations can be accomplished safely i.e. to set or increase
  + a key, add or remove elements from a set or increase a counter.
  + Redis supports a variety of languages i.e. C, C++, C#, Ruby, Python, Twisted Python, PHP, Erlang, Tcl, Perl, Lua, Java, Scala etc.
  + If your favorite language is not supported yet, you can write your own client library, as the Protocol is pretty simple.
  + Redis supports simple master to slave replication.

[PHP and Jquery Tutorial](https://www.wisdomjobs.com/e-university/php-and-jquery-tutorial-291.html" \o "PHP and Jquery Tutorial)

1. **Question 14. In Which Language Redis Is Written?**

**Answer :**

Redis is written in ANSI C and mostly used for cache solution and session management. It creates unique keys for store values.

1. **Question 15. What Are The Things You Must Have To Take Care While Using Redis?**

**Answer :**

**While using Redis, you must have to take care of following instructions:**

* + Select a consistent method to name and prefix your keys. Manage your namespace.
  + Create a “Registry” of key prefixes that maps each of your internal documents for those applications which “own” them.
  + For every class you put through into your Redis infrastructure: design, implement and test the mechanisms for garbage collection or data migration to archival storage.
  + Design, implement and test a sharding library before you have invested much into your application deployment and make sure that you keep a registry of “shards” replicated on each server.
  + Separate all your K/V store and related operations into your own library/API or service.

[PostgreSQL Interview Questions](https://www.wisdomjobs.com/e-university/postgresql-interview-questions.html" \o "PostgreSQL Interview Questions)

1. **Question 16. What Is The Difference Between Overriding A Value By Using Set Vs Append?**

**Answer :**

A value set by using SET command will set the value for the key.Doing it over again will override the value. However , using append has the effect of set only when the key has no value earlier, but if there is a value already assigned to the key , then doing append on the key would lead to an appended value to the existing value of the key.

[Django Tutorial](https://www.wisdomjobs.com/e-university/django-tutorial-1149.html" \o "Django Tutorial)

1. **Question 17. What Do You Mean By Data Modeling In Redis?**

**Answer :**

Just like any other database, data modeling represents the storage pattern of how and which data structures are used to store the data to achieve a domain requirement. For example in relational database world, we use primary key to establish a relationship between two entities. Data stored in relational databases are in table format, where as in Redis there are set of data structures available, which are used to represent the domain data. There is certainly different design mindset needed to convert the relational data to a Redis dataset.

[Master Data Management Interview Questions](https://www.wisdomjobs.com/e-university/master-data-management-interview-questions.html" \o "Master Data Management Interview Questions)

1. **Question 18. Why Redis Is Different As Compared To Other Key-value Stores?**

**Answer :**

* + Redis values can contain more complex data types, with atomic operations.
  + Redis is an in-memory but persistent on disk database.

[RDBMS Interview Questions](https://www.wisdomjobs.com/e-university/rdbms-interview-questions.html" \o "RDBMS Interview Questions)

1. **Question 19. How Do I Move A Redis Database From One Server To Another?**

**Answer :**

* + use BGSAVE command to Save a spanshot of the database into a dump.rdb
  + Copy this dump.rdb file into another server.

[Git (software) Tutorial](https://www.wisdomjobs.com/e-university/git-software-tutorial-1176.html" \o "Git (software) Tutorial)

1. **Question 20. How To Re-start Redis?**

**Answer :**

**You can restart Redis by using following path:**

/etc/init.d/redis-server restart

[Aerospace Interview Questions](https://www.wisdomjobs.com/e-university/aerospace-interview-questions.html" \o "Aerospace Interview Questions)

1. **Question 21. How To Start Redis?**

**Answer :**

**You can start Redis by using following path:**

/etc/init.d/redis-server start

1. **Question 22. How To Stop Redis?**

**Answer :**

You can stop Redis by using following path:  
/etc/init.d/redis-server stop

[Javascript Advanced Tutorial](https://www.wisdomjobs.com/e-university/javascript-advanced-tutorial-1214.html" \o "Javascript Advanced Tutorial)

1. **Question 23. How To Get All Keys From Redis?**

**Answer :**

**Use the following commands:**

$keyList = $redis->keys("\*");

print\_r($keyList);

/\*Array ( [0] => tutorials [1] => name ) \*/

[MYSQL DBA Interview Questions](https://www.wisdomjobs.com/e-university/mysql-dba-interview-questions.html" \o "MYSQL DBA Interview Questions)

1. **Question 24. How To Get Array Data From Redis?**

**Answer :**

**To get the array data from Redis use the following command** :

$list = $redis->lrange("tutorials", 0 ,5);     
print\_r($list);  
/\*Array ( [0] => Mysql [1] => Mongodb [2] => Redis [3] => MySQL [4] => PHP [5] => Mysql ) \*/

[PHP Interview Questions](https://www.wisdomjobs.com/e-university/php-interview-questions.html" \o "PHP Interview Questions)

1. **Question 25. How To Set Multiple Values (array) In Redis?**

**Answer :**

**To set multiple values in Redis use the following commands:**

$redis->lpush("tutorials", "PHP");  
$redis->lpush("tutorials", "MySQL");  
$redis->lpush("tutorials", "Redis");  
$redis->lpush("tutorials", "Mongodb");  
$redis->lpush("tutorials", "Mysql");

[Cucumber Tutorial](https://www.wisdomjobs.com/e-university/cucumber-tutorial-1407.html" \o "Cucumber Tutorial)

1. **Question 26. How To Get Value From Redis?**

**Answer :**

**To get value from Redis :**

echo $redis->get('name'); //Set string in redis

[Ruby Interview Questions](https://www.wisdomjobs.com/e-university/ruby-interview-questions.html" \o "Ruby Interview Questions)

1. **Question 27. How To Set Value In Redis?**

**Answer :**

**To set the value in Redis use the following command:**

$redis->set("name", "Set string in redis");

[MySQL Interview Questions](https://www.wisdomjobs.com/e-university/mysql-interview-questions.html" \o "MySQL Interview Questions)

1. **Question 28. How To Check Redis Is Running?**

**Answer :**

**To check Redis is running try out the following code :**

try  
{  
    $redis = new Redis();  
    $redis->connect('127.0.0.1', 6379);  
    echo "Redis is running.";  
    echo "Server is running: " . $redis->ping();  
}  
catch (Exception $e)   
{  
    echo $e->getMessage();  
}

1. **Question 29. How To Remove All Database?**

**Answer :**

**To remove all database use the following code:**

redis-cli flushall

[Django Interview Questions](https://www.wisdomjobs.com/e-university/django-interview-questions.html" \o "Django Interview Questions)

1. **Question 30. How To Delete Current Database?**

**Answer :**

**To delete current database use the following code:**

redis-cli flushdb

1. **Question 31. What Do You Mean By "redis Is Binary Safe"?**

**Answer :**

Binary safe means that it has a known length but not limited by any special character. You can store any value upto the given size. A string value can be 512 MB in length.

1. **Question 32. How Does Redis Differ From Mongodb? Is There A Use Case When For Redis If Using Mongodb?**

**Answer :**

MongoDB is an unstructured (at least, in definition) distributed document storage. For clearing purpose you can think about it as a high scalable and not so high performance JSON storage (though actually it uses BSON). It has a simple but effective multitype index system, replication and sharding, and a lot of more nice features.

Redis is a simplest key/value store, with transient data caching. It can works as an efficient queue system.

Looking for more specific information you will find a lot of scenarios where both them are used, Mongo as a true persistent canonical storage, and Redis as an transient/cache storage.

**1. List some Redis Clients supported by PHP?**

Below are some Redis Clients supported by [PHP Programming Language](https://www.onlineinterviewquestions.com/core-php-interview-questions/)

* amphp/redis
* cheprasov/php-redis-client
* Credis
* PHP Redis implementation / wrapper
* PHP Sentinel Client
* phpredis

**2. What is redis-cli.**

redis-cli is the Redis command line interface, a simple program that allows to send commands to Redis, and read the replies sent by the server, directly from the terminal.

**3. Explain REPL**

REPL stands for Read Eval Print Loop. It an interactive mode where the user types commands and get replies.

**4. List the programming languages supported by Redis?**

Redis supports a wide range of programming language. Some major programming languages supported by Redis are PHP, Java, Python, Scala, Perl, Ruby, C#, and C++.

**5. What is Redis?**

Redis is an open source released under BSD licensed, in-memory data structure store, it can be used as a database, cache and message broker.

**6. List the data structures supported by Redis.**

Redis supports following Data Structures

* Strings
* Hashes
* Lists
* Sets
* Sorted sets with range queries
* bitmaps
* Hyperloglogs
* Geospatial indexes with radius queries

**7. List some advantages of Redis?**

Following are some advantages of Redis

* Exceptionally Faster than [others](https://www.onlineinterviewquestions.com/others/)
* Supports for rich data types like Hashes, Sets, bitmaps
* Rich client-side library.
* Support for server-side locking.
* Operations are atomic.

**8. List some commonly used Redis commands?**

Some commonly used Redis commands list

| **Command Name** | **Description** |
| --- | --- |
| APPEND | Append a value to a key |
| AUTH | Authenticate to the server |
| BGREWRITEAOF | Asynchronously rewrite the append-only file |
| BGSAVE | Asynchronously save the dataset to disk |
| BITCOUNT | Count set bits in a string |
| CLIENT LIST | Get the list of client connections |
| CLUSTER INFO | Provides info about Redis Cluster node state |

Read more from [Redis Commands List](https://redis.io/commands)

**9. In which language Redis is written?**

Redis is [NoSql](https://www.onlineinterviewquestions.com/nosql-interview-questions-answers/)based Key-value Database, which is written in **ANSI C**

Interview Questions on Redis for Developers

[](https://medium.com/@cosmicconvallis?source=post_page---------------------------)

[Nitish Prabhu](https://medium.com/@cosmicconvallis?source=post_page---------------------------)

Follow

[Jun 9, 2018](https://medium.com/@cosmicconvallis/interview-questions-on-redis-for-developers-c27769b410e9?source=post_page---------------------------) · 7 min read

**1. What is Redis?**

[Redis](https://www.amazon.in/Redis-Action-Josiah-L-Carlson/dp/1617290858/ref=as_li_ss_tl?ie=UTF8&qid=1526492268&sr=8-2&keywords=redis&linkCode=ll1&tag=nitishprabhu-21&linkId=10ada029ae249728dd58318d873719e7&source=post_page---------------------------)is an open source, in-memory data structure store, used as a database, cache and message broker. It supports data structures such as strings, hashes, lists, sets, sorted sets with range queries, bitmaps, hyperloglogs and geospatial indexes with radius queries.

**2. What is in-memory Database?**

The in-memory database is a database where it keeps the dataset in RAM. Means that for every interaction with database, you will only access the Main memory; No DISK operations involved during this interaction. Hence the operation will be faster as it directly aceess main memoery instead of any disk operation.

**3. Isn’t the Redis data lost if there’s a system crash?**

To avoid data loss in case of system crash Redis provides persistence by Snapshotting & AOF(Append-only files).

**4. How Redis achieve persistence by Snapshotting?**

Snapshot takes the data as it exists at one moment in time & writes it to the disk, will be written to the file referenced as “**dbfilename**”.

*There are 5 steps to initiate snapshots.*

a. BGSAVE: When redis client initiate BGSAVE, redis will create a fork so that child process will write a the snapshot to the disk where master interacts with commands.

b. SAVE: When redis client initiate SAVE, redis will stop responding any commands until snapshot completes

c. From Configuration Files:

Example: Configuration as SAVE 60 10000

BGSAVE will be called if 10000 writes occure within 60 seconds

d. SHUTDOWN: When redis client sends a SHUTDOWN, redis will perform SAVE operations, blocks all operations, shutdowns later.

e. Sync with other Redis: If a Redis server connects to another server, it issues SYNC, then master Redis will start BGSAVE.

**5. How redis achive persistence by AOF?**

AOF copies incoming write command to disk as they happen. It provides file synch options as follows.

a. always — Every writes to redis, writes to disk which usually affect redis’ performance.

b. everysec — Writes to disk every second

c. no — Let OS control syncing to disk

**6. Files will keep on growing as long as AOF executed. Won’t it consume more memory? What is the solution provided by redis?**

Redis provides command BGREWRITEAOF which rewrites the existing AOF with the shortest sequence of commands needed to rebuild the current dataset in memory.

**7. Redis String data-type.**

Redis Strings are binary safe, this means that a Redis string can contain any kind of data, for instance a JPEG image or a serialized Ruby object.

A String value can be at max 512 Megabytes in length.

**8. Explain SET & GET command.**

SET command used to set a value to the key and GET command to get a value of the key as follows, of data-type STRING.

*127.0.0.1:6379> SET TEST\_KEY TEST\_VALUE*

*OK [TEST\_VALUE is being set to TEST\_KEY]*

*127.0.0.1:6379> GET TEST\_KEY*

*"TEST\_VALUE"*

SET Command with NX|XX arguments

NX argument indicates that SET a value to the key if the key is not exist;

XX argument indicates that SET a value to the key only if the key already exists;

Example:

NX: key TEST\_KEY is already exist; If we try to update TEST\_KEY value, it will not accept and returns (nill) status

*127.0.0.1:6379> SET TEST\_KEY TEST\_VALUE\_UPDATED NX*

*(nil)*

*127.0.0.1:6379> GET TEST\_KEY*

*"TEST\_VALUE"*

Value of TEST\_KEY remain same;

XX: key TEST\_KEY is alrady exists; Hence so, when we try to udpate TEST\_KEY value, its updated and ack OK

*127.0.0.1:6379> SET TEST\_KEY TEST\_VALUE\_XX\_UDPATED XX*

*OK*

*127.0.0.1:6379> GET TEST\_KEY*

*"TEST\_VALUE\_XX\_UDPATED"*

**9. Redis List data-type.**

Internal Implementation: Linked-List

Redis Lists are simply lists of strings, sorted by insertion order. It is possible to add elements to a Redis List pushing new elements on the head (on the left) or on the tail (on the right) of the list.

The max length of a list is 2³² — 1 elements (4294967295, more than 4 billion of elements per list).

**10. Explain List operations.**

LPUSH — Inserts a new element on the head

RPUSH — Inserts a new element on the tail

LPOP — Removes an element from the head

RPOP — Removes an element from tail

LRANGE — Prints an element from the specified position

Example:

*127.0.0.1:6379> LPUSH A 1*

*127.0.0.1:6379> RPUSH A 2*

*127.0.0.1:6379> RPUSH A 3*

*127.0.0.1:6379> RPUSH A 4*

*127.0.0.1:6379> LRANGE A 0 -1*

*1) "1"*

*2) "2"*

*3) "3"*

*4) "4"*

*127.0.0.1:6379> LPOP A*

*"1" ["1" element value is removed]*

*127.0.0.1:6379> LRANGE A 0 -1*

*1) "2"*

*2) "3"*

*3) "4"*

*127.0.0.1:6379> RPOP A*

*"4" ["4" element value is removed]*

*127.0.0.1:6379> LRANGE A 0 -1*

*1) "2"*

*2) "3"*

**11. Explain Blocking operations on lists.**

This is a case when you want to do remove operations like RPOP/LPOP, but the list is already empty. In this case, remove operation will return null does nothing. In this case a consumer is forced to wait some time and retry again with RPOP.

So Redis implements commands called BRPOP and BLPOP which are versions of RPOP and LPOP able to block if the list is empty: This will return to the caller only when a new element is added to the list, or when a user-specified timeout is reached.

Example: If we want pop the elements from the list A, which is empty right now;

*127.0.0.1:6379> brpop A {timeout}*

*...*

*...*

waits until new elements is added to list key A

timeout — No of seconds to wait until; 0 indicates wait forever

**12. Redis SET data-type.**

Redis Sets are an unordered collection of Strings. It is possible to add, remove, and test for existence of members in O(1)

The max number of members in a set is 2³² — 1 (4294967295, more than 4 billion of members per set).

**13. Explain Redis SET operations.**

SADD — Adds a new element to the SET

SMEMBERS — To retrieve values from the KEY

SISMEMBER — To verify whether specified element is present in the SET

SPOP — To pop the element out from the SET

Example:

*127.0.0.1:6379> SADD TEST\_SET A*

*(integer) 1*

*127.0.0.1:6379> SADD TEST\_SET B*

*(integer) 1*

*127.0.0.1:6379> SMEMBERS TEST\_SET*

*1) "A"*

*2) "B"*

*127.0.0.1:6379> SISMEMBER TEST\_SET A*

*(integer) 1 [1 indicates it's true; it's available]*

*127.0.0.1:6379> SPOP TEST\_SET*

*"A"*

*127.0.0.1:6379> SMEMBERs TEST\_SET*

*1) "B"*

**14. Explain SINTER & SUNION.**

SUNION — Returns the members of the set resulting from the union of all the given sets.

SINTER — Returns the members of the set resulting from the intersection of all the given sets.

Example:

*127.0.0.1:6379> SMEMBERS TEST\_KEY\_A*

*1) "1"*

*2) "2"*

*3) "3"*

*127.0.0.1:6379> SMEMBERS TEST\_KEY\_B*

*1) "2"*

*2) "6"*

*127.0.0.1:6379> SUNION TEST\_KEY\_A TEST\_KEY\_B*

*1) "1"*

*2) "2"*

*3) "3" [prints union of two sets]*

*4) "6"*

*127.0.0.1:6379> SINTER TEST\_KEY\_A TEST\_KEY\_B*

*1) "2" [print intersection of two sets]*

**15. Explain SUNIONSTORE & SINTERSTORE.**

SUNIONSTORE — This command is equal to SUNION, but instead of returning the resulting set, it is stored in destination.

SINTERSTORE — This command is equal to SINTER, but instead of returning the resulting set, it is stored in destination.

Example:

*127.0.0.1:6379> SUNIONSTORE TEST\_KEY\_UNION TEST\_KEY\_A TEST\_KEY\_B*

*(integer) 4*

*127.0.0.1:6379> SMEMBERS TEST\_KEY\_UNION*

*1) "1"*

*2) "2"*

*3) "3"*

*4) "6"*

*127.0.0.1:6379> SINTERSTORE TEST\_KEY\_INTER TEST\_KEY\_A TEST\_KEY\_B*

*(integer) 1*

*127.0.0.1:6379> SMEMBERS TEST\_KEY\_INTER*

*1) "2"*

**16. Redis Hashes data-type.**

Redis Hashes are maps between string fields and string values, so they are the perfect data type to represent objects like user-details

Every hash can store up to 2³² — 1 field-value pairs (more than 4 billion).

**17. Explain HASH data-type operation.**

HMSET — Sets the specified fields to their respective values in the hash stored at key.

*Time complexity:* O(N) where N is the number of fields being set.

HGET — Returns the value associated with field in the hash stored at key.

*Time complexity:* O(1)

HGETALL — Returns all fields and values of the hash stored at key.

*Time complexity:* O(N) where N is the size of the hash.

Example:

*127.0.0.1:6379> HMSET TEST\_HASH\_KEY A 20*

*OK*

*127.0.0.1:6379> HMSET TEST\_HASH\_KEY B 10*

*OK*

*127.0.0.1:6379> HMSET TEST\_HASH\_KEY C 10*

*OK*

*127.0.0.1:6379> HGETALL TEST\_HASH\_KEY*

*1) "A"*

*2) "20"*

*3) "B"*

*4) "10"*

*5) "C"*

*6) "10"*

*127.0.0.1:6379> HMGET TEST\_HASH\_KEY A*

*1) "20"*

**18. Explain Redis Sorted sets.**

Sorted sets are a data type which is similar to a mix between a Set and a Hash. Like sets, sorted sets are composed of unique, non-repeating string elements, so in some sense a sorted set is a set as well.

However while elements inside sets are not ordered, every element in a sorted set is associated with a floating point value, called the score.[in ascending order]

**19. Explain redis Sorted Set operation.**

ZADD — Adds all the specified members with the specified scores to the sorted set stored at key.

*Time complexity:* O(log(N)) for each item added, where N is the number of elements in the sorted set.

ZREM — Removes the specified members from the sorted set stored at key. Non existing members are ignored.

*Time complexity:* O(M\*log(N)) with N being the number of elements in the sorted set and M the number of elements to be removed.

ZRANGE — Returns the specified range of elements in the sorted set stored at key.

*Time complexity:* O(log(N)+M) with N being the number of elements in the sorted set and M the number of elements returned.

ZRANGEBYSCORE — Returns all the elements in the sorted set at key with a score between min and max.

*Time complexity*: O(log(N)+M) with N being the number of elements in the sorted set and M the number of elements being returned. If M is constant (e.g. always asking for the first 10 elements with LIMIT), you can consider it O(log(N)).

*127.0.0.1:6379> ZADD TEST\_ZKEY 1 A*

*(integer) 1*

*127.0.0.1:6379> ZADD TEST\_ZKEY 2 B*

*(integer) 1*

*127.0.0.1:6379> ZADD TEST\_ZKEY 1 C*

*(integer) 1*

*127.0.0.1:6379> ZRANGE TEST\_ZKEY 0 -1*

*1) "A"*

*2) "C"*

*3) "B"*

*127.0.0.1:6379> ZREM TEST\_ZKEY B*

*(integer) 1*

*127.0.0.1:6379> ZRANGE TEST\_ZKEY 0 -1*

*1) "A"*

*2) "C"*

*127.0.0.1:6379> ZRANGEBYSCORE TEST\_ZKEY 1 2 {Here 1 is min rank, 2 is max rank- gets a data from rank of 1 to 2}*

*1) "A"*