<https://www.tutorialspoint.com/mongodb/index.htm>

<https://www.w3schools.com/nodejs/nodejs_mongodb.asp>

MongoDB

A MongoDB egy nyít forráskódú NoSQL adatbázis.

Minta dokumentum:

{

\_id: ObjectId(7df78ad8902c)

title: 'MongoDB Overview',

description: 'MongoDB is no sql database',

by: 'tutorials point',

url: 'http://www.tutorialspoint.com',

tags: ['mongodb', 'database', 'NoSQL'],

likes: 100,

comments: [

{

user:'user1',

message: 'My first comment',

dateCreated: new Date(2011,1,20,2,15),

like: 0

},

{

user:'user2',

message: 'My second comments',

dateCreated: new Date(2011,1,25,7,45),

like: 5

}

]

}

Minden relációs adatbázisnak van egy tipikus sémája, amely bemutatja a táblák számát és a táblák közötti kapcsolatot. Míg a MongoDB-ben nincs ilyen kapcsolati fogalom.

**A MongoDB előnyei az RDBMS-hez képest**

Kevesebb séma - A MongoDB egy olyan dokumentum-adatbázis, amelyben a gyűjtemények különböző dokumentumokat tárolnak. A mezők száma, tartalma és a dokumentum mérete dokumentumonként eltérő lehet.

Objektum struktúra

Nincs komplex join

A MongoDB támogatja a dokumentumok dinamikus lekérdezését olyan dokumentum-alapú lekérdezési nyelv használatával, amely majdnem olyan hatékony, mint az SQL.

Az alkalmazásobjektumok konvertálása / leképezése adatbázis-objektumokká nem szükséges.

**Miért jó a MongoDB?**

Dokumentum-orientált tárolás - Az adatokat JSON-stílusú dokumentumok formájában tárolják.

Index bármely attribútumhoz

Replikáció és magas rendelkezésre állás

Gazdag lekérdezés

**Hol használható a MongoDB?**

Nagy adatsor

Tartalomkezelés

Felhasználói adatok kezelése, stb

Adatbázis létrehozása

var MongoClient = require('mongodb').MongoClient;  
var url = "mongodb://localhost:27017/mydb";  
  
MongoClient.connect(url, function(err, db) {  
  if (err) throw err;  
  console.log("Database created!");  
  db.close();  
});

Collection létrehozása

var MongoClient = require('mongodb').MongoClient;  
var url = "mongodb://localhost:27017/";  
  
MongoClient.connect(url, function(err, db) {  
  if (err) throw err;  
  var dbo = db.db("mydb");  
  dbo.createCollection("customers", function(err, res) {  
    if (err) throw err;  
    console.log("Collection created!");  
    db.close();  
  });  
});

Node.js MongoDB Insert

## Insert Into Collection

var MongoClient = require('mongodb').MongoClient;  
var url = "mongodb://localhost:27017/";  
  
MongoClient.connect(url, function(err, db) {  
  if (err) throw err;  
  var dbo = db.db("mydb");  
  var myobj = { name: "Company Inc", address: "Highway 37" };  
  dbo.collection("customers").insertOne(myobj, function(err, res) {  
    if (err) throw err;  
    console.log("1 document inserted");  
    db.close();  
  });  
});

## Insert Multiple Documents

var MongoClient = require('mongodb').MongoClient;  
var url = "mongodb://localhost:27017/";  
  
MongoClient.connect(url, function(err, db) {  
  if (err) throw err;  
  var dbo = db.db("mydb");  
  var myobj = [  
    { name: 'John', address: 'Highway 71'},  
    { name: 'Peter', address: 'Lowstreet 4'},  
    { name: 'Amy', address: 'Apple st 652'},  
    { name: 'Hannah', address: 'Mountain 21'},  
    { name: 'Michael', address: 'Valley 345'},  
    { name: 'Sandy', address: 'Ocean blvd 2'},  
    { name: 'Betty', address: 'Green Grass 1'},  
    { name: 'Richard', address: 'Sky st 331'},  
    { name: 'Susan', address: 'One way 98'},  
    { name: 'Vicky', address: 'Yellow Garden 2'},  
    { name: 'Ben', address: 'Park Lane 38'},  
    { name: 'William', address: 'Central st 954'},  
    { name: 'Chuck', address: 'Main Road 989'},  
    { name: 'Viola', address: 'Sideway 1633'}  
  ];  
  dbo.collection("customers").insertMany(myobj, function(err, res) {  
    if (err) throw err;  
    console.log("Number of documents inserted: " + res.insertedCount);  
    db.close();  
  });  
});

## The \_id Field

Ha nem adunk meg \_id mezőt, akkor a MongoDB elkészíti, tehát a MongoDB minden dokumentumhoz egyedi azonosítót rendel.

Ha megadjuk az \_id mezőt, akkor az értéknek egyedinek kell lennie:

var MongoClient = require('mongodb').MongoClient;  
var url = "mongodb://localhost:27017/";  
  
MongoClient.connect(url, function(err, db) {  
  if (err) throw err;  
  var dbo = db.db("mydb");  
  var myobj = [  
    { **\_id: 154**, name: 'Chocolate Heaven'},  
    { **\_id: 155**, name: 'Tasty Lemon'},  
    { **\_id: 156**, name: 'Vanilla Dream'}  
  ];  
  dbo.collection("products").insertMany(myobj, function(err, res) {  
    if (err) throw err;  
    console.log(res);  
    db.close();  
  });  
});

Node.js MongoDB Find

## Find One

A MongoDB gyűjteményéből adatok kiválasztásához a findOne() metódust használhatjuk.

A findOne() metódus visszaadja a kiválasztás első előfordulását.

A findOne() metódus első paramétere egy lekérdezési objektum. Ebben a példában egy üres lekérdezési objektumot használunk, amely kiválasztja a gyűjtemény összes dokumentumát (de csak az első dokumentumot adja vissza).

var MongoClient = require('mongodb').MongoClient;  
var url = "mongodb://localhost:27017/";  
  
MongoClient.connect(url, function(err, db) {  
  if (err) throw err;  
  var dbo = db.db("mydb");  
  dbo.collection("customers").findOne({}, function(err, result) {  
    if (err) throw err;  
    console.log(result.name);  
    db.close();  
  });  
});

## Find All

Az adatok kiválasztásához a find() függvényt használjuk.

A find() visszaadja a kiválasztás összes előfordulását.

A find() első paramétere egy lekérdezési objektum. Ebben a példában egy üres lekérdezési objektumot használunk, amely kiválasztja a gyűjtemény összes dokumentumát.

var MongoClient = require('mongodb').MongoClient;  
var url = "mongodb://localhost:27017/";  
  
MongoClient.connect(url, function(err, db) {  
  if (err) throw err;  
  var dbo = db.db("mydb");  
  dbo.collection("customers").find({}).toArray(function(err, result) {  
    if (err) throw err;  
    console.log(result);  
    db.close();  
  });  
});

## Find Some

A find() második paramétere egy objektum, amely leírja, mely mezőket szeretnénk látni az eredménynél

Ez a paraméter nem kötelező, ha nem adjuk meg akkor minden mezőt vissszaad.

var MongoClient = require('mongodb').MongoClient;  
var url = "mongodb://localhost:27017/";  
  
MongoClient.connect(url, function(err, db) {  
  if (err) throw err;  
  var dbo = db.db("mydb");  
  dbo.collection("customers").find({},**{ projection: { \_id: 0, name: 1, address: 1 } }**).toArray(function(err, result) {  
    if (err) throw err;  
    console.log(result);  
    db.close();  
  });  
});

var MongoClient = require('mongodb').MongoClient;  
var url = "mongodb://localhost:27017/";  
  
MongoClient.connect(url, function(err, db) {  
  if (err) throw err;  
  var dbo = db.db("mydb");  
  dbo.collection("customers").find({},**{ projection: { address: 0 } }**).toArray(function(err, result) {  
    if (err) throw err;  
    console.log(result);  
    db.close();  
  });  
});

var MongoClient = require('mongodb').MongoClient;  
var url = "mongodb://localhost:27017/";  
  
MongoClient.connect(url, function(err, db) {  
  if (err) throw err;  
  var dbo = db.db("mydb");  
  dbo.collection("customers").find({},**{ projection: { \_id: 0, name: 1 } }**).toArray(function(err, result) {  
    if (err) throw err;  
    console.log(result);  
    db.close();  
  });  
});

var MongoClient = require('mongodb').MongoClient;  
var url = "mongodb://localhost:27017/";  
  
MongoClient.connect(url, function(err, db) {  
  if (err) throw err;  
  var dbo = db.db("mydb");  
  dbo.collection("customers").find({},**{ projection: { \_id: 0 } }**).toArray(function(err, result) {  
    if (err) throw err;  
    console.log(result);  
    db.close();  
  });  
});

Node.js MongoDB Query

## Filter the Result

A talált eredményeket egy lekérdezési objektum segítségével szűrhetjük.

A find() első argumentuma egy lekérdezési objektum, és a keresés korlátozására szolgál.

var MongoClient = require('mongodb').MongoClient;  
var url = "mongodb://localhost:27017/";  
  
MongoClient.connect(url, function(err, db) {  
  if (err) throw err;  
  var dbo = db.db("mydb");  
**var query = { address: "Park Lane 38" };**  dbo.collection("customers").find(**query**).toArray(function(err, result) {  
    if (err) throw err;  
    console.log(result);  
    db.close();  
  });  
});

## Filter With Regular Expressions

var MongoClient = require('mongodb').MongoClient;  
var url = "mongodb://localhost:27017/";  
  
MongoClient.connect(url, function(err, db) {  
  if (err) throw err;  
  var dbo = db.db("mydb");  
  var query = { address: **/^S/** };  
  dbo.collection("customers").find(query).toArray(function(err, result) {  
    if (err) throw err;  
    console.log(result);  
    db.close();  
  });  
});

Node.js MongoDB Sort

## Sort the Result

A sort() rendezi az eredményt növekvő vagy csökkenő sorrendben.

A sort()-nak egy paramétere van, meghatározza a rendezési sorrendet.

var MongoClient = require('mongodb').MongoClient;  
var url = "mongodb://localhost:27017/";  
  
MongoClient.connect(url, function(err, db) {  
  if (err) throw err;  
  var dbo = db.db("mydb");  
**var mysort = { name: 1 };**  dbo.collection("customers").find()**.sort(mysort)**.toArray(function(err, result) {  
    if (err) throw err;  
    console.log(result);  
    db.close();  
  });  
});

## Sort Descending

A -1 érték használatával csökkenő sorrendbe rendezhetünk.

var MongoClient = require('mongodb').MongoClient;  
var url = "mongodb://localhost:27017/";  
  
MongoClient.connect(url, function(err, db) {  
  if (err) throw err;  
  var dbo = db.db("mydb");  
**var mysort = { name: -1 };**  dbo.collection("customers").find()**.sort(mysort)**.toArray(function(err, result) {  
    if (err) throw err;  
    console.log(result);  
    db.close();  
  });  
});

Node.js MongoDB Delete

## Delete Document

Egy rekord vagy dokumentum törléséhez, a deleteOne() metódust használjuk.

A deleteOne() első paramétere egy objektum, amely meghatározza, hogy melyik dokumentumot kell törölni.

var MongoClient = require('mongodb').MongoClient;  
var url = "mongodb://localhost:27017/";  
  
MongoClient.connect(url, function(err, db) {  
  if (err) throw err;  
  var dbo = db.db("mydb");  
**var myquery = { address: 'Mountain 21' };**  dbo.collection("customers").deleteOne(myquery, function(err, obj) {  
    if (err) throw err;  
    console.log("1 document deleted");  
    db.close();  
  });  
});

## Delete Many

Egynél több dokumentum törlésére a deleteMany() szolgál.

A deleteMany() első paramétere egy objektum, amely meghatározza, hogy mely dokumentumokat töröljük.

var MongoClient = require('mongodb').MongoClient;  
var url = "mongodb://localhost:27017/";  
  
MongoClient.connect(url, function(err, db) {  
  if (err) throw err;  
  var dbo = db.db("mydb");  
**var myquery = { address: /^O/ };**  dbo.collection("customers").deleteMany(myquery, function(err, obj) {  
    if (err) throw err;  
    console.log(obj.result.n + " document(s) deleted");  
    db.close();  
  });  
});

## The Result Object

A deleteMany() olyan objektumot ad vissza, amely információkat tartalmaz arról, hogy a végrehajtás hogyan befolyásolta az adatbázist.

Azt adja meg, hogy a végrehajtás rendben volt-e, és hány dokumentumot érintett.

Node.js MongoDB Drop

## Drop Collection

A drop() használatával törölhetünk egy táblát vagy gyűjteményt.

A drop() callback függvény, amely tartalmazza a hibaobjektumot és az eredményt, amely true értékat ad vissza, ha a gyűjtemény sikeresen törölve lett, különben false értéket ad vissza.

var MongoClient = require('mongodb').MongoClient;  
var url = "mongodb://localhost:27017/";  
  
MongoClient.connect(url, function(err, db) {  
  if (err) throw err;  
  var dbo = db.db("mydb");  
  dbo.collection("customers").drop(function(err, delOK) {  
    if (err) throw err;  
    if (delOK) console.log("Collection deleted");  
    db.close();  
  });  
});

## db.dropCollection

A dropCollection() segítségével is törölhetünk egy táblát (gyűjteményt).

A dropCollection()-nek két paramétere van, gyűjtemény nevét és a callback függvényt jelenti.

var MongoClient = require('mongodb').MongoClient;  
var url = "mongodb://localhost:27017/";  
  
MongoClient.connect(url, function(err, db) {  
  if (err) throw err;  
  var dbo = db.db("mydb"); **dbo.dropCollection("customers", function(err, delOK) {**    if (err) throw err;  
    if (delOK) console.log("Collection deleted");  
    db.close();  
  });  
});

Node.js MongoDB Update

## Update Document

Frissíthetünk egy rekordot vagy dokumentumot az updateOne() függvénnyel.

Az updateOne() első paramétere egy lekérdezési objektum, amely meghatározza, hogy melyik dokumentumot kell frissíteni.

var MongoClient = require('mongodb').MongoClient;  
var url = "mongodb://127.0.0.1:27017/";  
  
MongoClient.connect(url, function(err, db) {  
  if (err) throw err;  
  var dbo = db.db("mydb");  
  var myquery = { address: "Valley 345" };  
  var newvalues = { $set: {name: "Mickey", address: "Canyon 123" } };  
  dbo.collection("customers").updateOne(myquery, newvalues, function(err, res) {  
    if (err) throw err;  
    console.log("1 document updated");  
    db.close();  
  });  
});

## Update Only Specific Fields

A $ set operátor használatakor csak a megadott mezők frissülnek:

var MongoClient = require('mongodb').MongoClient;

var url = "mongodb://localhost:27017/";

MongoClient.connect(url, function(err, db) {

if (err) throw err;

var dbo = db.db("mydb");

var myquery = { address: "Valley 345" };

var newvalues = {$set: {address: "Canyon 123"} };

dbo.collection("customers").updateOne(myquery, newvalues, function(err, res) {

if (err) throw err;

console.log("1 document updated");

db.close();

});

});

## Update Many Documents

Az összes dokumentum frissítéséhez, amely megfelel a lekérdezés feltételeinek, az updateMany() függvényt használjuk.

var MongoClient = require('mongodb').MongoClient;  
var url = "mongodb://127.0.0.1:27017/";  
  
MongoClient.connect(url, function(err, db) {  
  if (err) throw err;  
  var dbo = db.db("mydb");  
  var myquery = { address: /^S/ };  
  var newvalues = {$set: {name: "Minnie"} };  
  dbo.collection("customers").updateMany(myquery, newvalues, function(err, res) {  
    if (err) throw err;  
    console.log(res.result.nModified + " document(s) updated");  
    db.close();  
  });  
});

Node.js MongoDB Limit

## Limit the Result

Az eredmény korlátozására a MongoDB-ben a limit() függvényt használjuk.

A limit() függvénynek egy paramétere van, egy szám.

var MongoClient = require('mongodb').MongoClient;  
var url = "mongodb://localhost:27017/";  
  
MongoClient.connect(url, function(err, db) {  
  if (err) throw err;  
  var dbo = db.db("mydb");  
  dbo.collection("customers").find()**.limit(5)**.toArray(function(err, result) {  
    if (err) throw err;  
    console.log(result);  
    db.close();  
  });  
});

Node.js MongoDB Join

## Join Collections

A MongoDB nem egy relációs adatbázis, de a join-t a $ lookup segítségével hajthatja végre.

A $ lookup lehetővé teszi annak meghatározását, hogy melyik gyűjteményt szeretnék a join-ba bevonni az aktuális gyűjteményhez, és mely mezőknek kell egyezniük a gyűjtemények esetén.

var MongoClient = require('mongodb').MongoClient;  
var url = "mongodb://127.0.0.1:27017/";  
  
MongoClient.connect(url, function(err, db) {  
  if (err) throw err;  
  var dbo = db.db("mydb");  
  dbo.collection('orders').aggregate([  
**{ $lookup:  
       {  
         from: 'products',  
         localField: 'product\_id',  
         foreignField: '\_id',  
         as: 'orderdetails'  
       }  
     }**    ]).toArray(function(err, res) {  
    if (err) throw err;  
    console.log(JSON.stringify(res));  
    db.close();  
  });  
});