

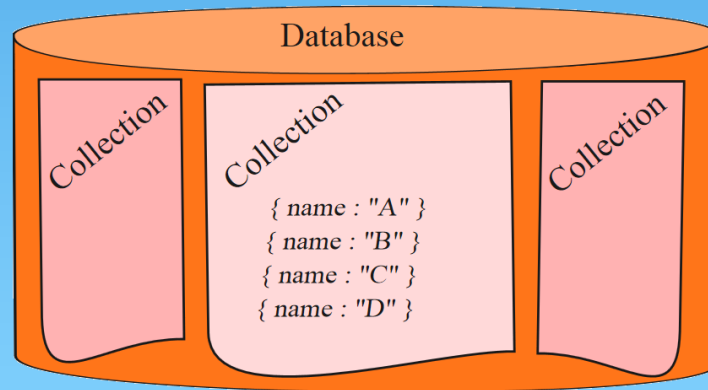
# הנדסת תוכנה

## פיתוח צד שרת



# mongoDB

mongoDB = “Humongous DB”



Our dataBase!

# מבוא ל- MongoDB

\* MongoDB הוא מסד נתונים לא-רלציוני בקוד פתוח (formerly 10gen)

\* Document-based

“High performance, high availability”

\* Automatic scaling

\* JSON syntax like (BSON) מורכב כזוג שדה-ערך

\* שפת שאילתה עשירה

\* ממשק קל עם שפות נפוצות (Java, JavaScript, PHP וכו')

\* ניתן להרצה בכל (VM's, cloud וכו')

<https://www.mongodb.org>

\* דוקומנטציה, התקנה, הדרכה ועוד בלינק-



# noSQL, none relational Db

\* כיום בפלטפורמות רבות, נהוג להשתמש במסדי נתונים שאינם ראציונלים ואינם SQL ים

\* הסיבות הן פשוטות:

\* רובנו לא משתמשים במסד נתונים כל כך גדול שאנחנו צריכים את היכולות של SQL

\* רובנו לא באמת עושים חתכים מיוחדים בנתונים ולא מערבבים ביניהם הרבה

\* כולנו רוצים דרך נוחה וקלה יותר לבנות את מסד הנתונים

\* נעדיף שזה ירגיש כמו תכנות, ויהיה חלק מהשפה, עם אותו הסינטקס ואותה ההתנהגות

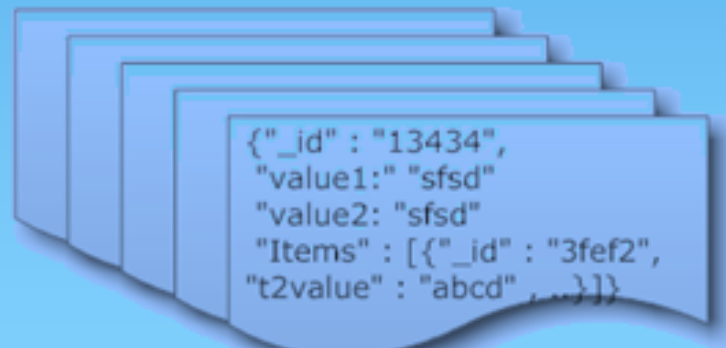
\* ולכן, כולנו משתמשים במסדי נתונים שאינם SQL לרוב, כמו MongoDB

# Document Database

- \* מבוסס מסמך, המסמך דומה לאובייקט JSON
- \* כל מסמך מאחסן אוסף, מורכב מזוג "שדה-ערך" (key: value)
- \* הערכים עשויים להכיל מסמכים אחרים, מערכים או מערכים של מסמכים

## Document Model

Collection ("Things")



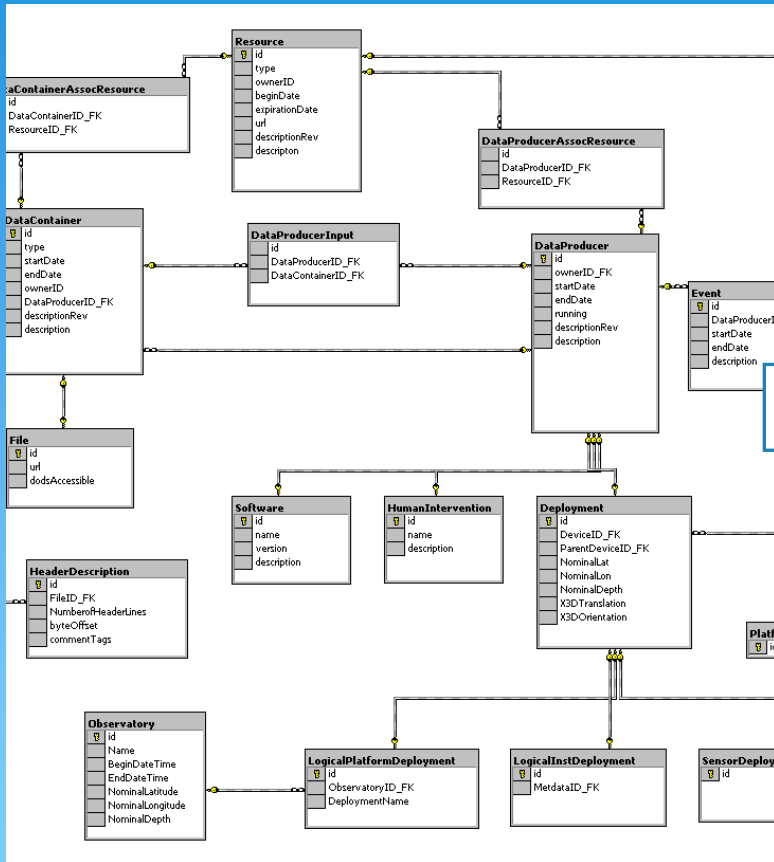
\* אוסף / collection:

- \* תמיכת אינדקס
- \* בדומה לטבלאות ב-DB
- \* לא נדרש מבנה אחיד

# Collections

- \* במקום טבלאות, אנחנו עובדים עם אוספים.
- \* אוסף מייצג בשבילנו מאגר נתונים עם אופי אחיד. כמו רשימת משתמשים
- \* אוסף יכול להכיל מסמכים
- \* אין שום הגבלה או התנייה למבנה המסמכים, אוסף יכול הלכי מסמכים עם מבנה שונה
- \* את פעולות החיפוש, סינון, וכל השאר נבצע על אוסף

# Document Model VS Relational Model



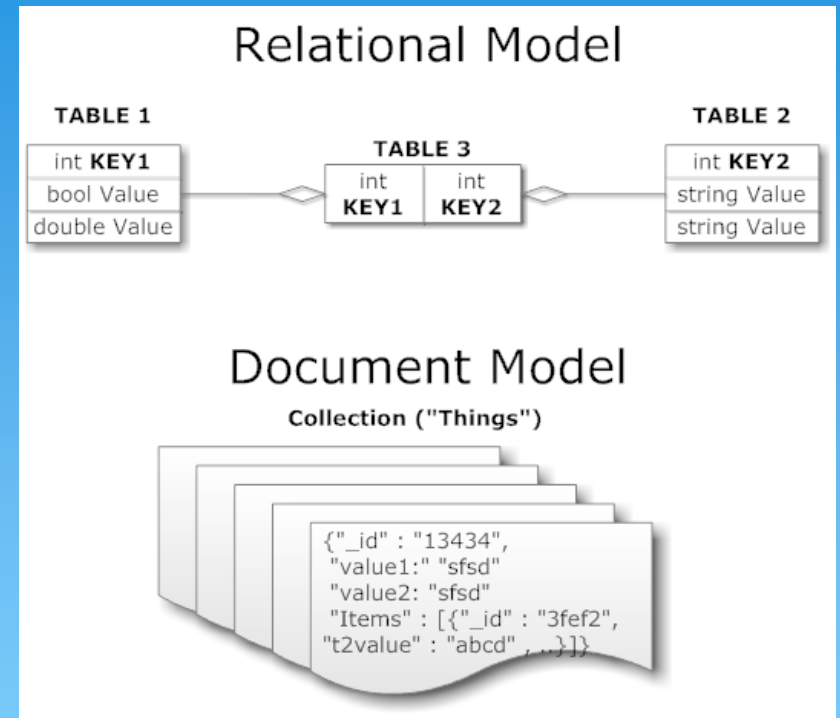
```

{
  title: 'MongoDB',
  contributors: [
    { name: 'Eliot Horowitz',
      email: 'eliot@10gen.com' },
    { name: 'Dwight Merriman',
      email: 'dwight@10gen.com' }
  ],
  model: {
    relational: false,
    awesome: true
  }
}

```

# Document Model VS Relational Model

RDBMS		MongoDB
Database	➡	Database
Table, View	➡	Collection
Row	➡	Document (JSON, BSON)
Column	➡	Field
Index	➡	Index
Join	➡	Embedded Document
Foreign Key	➡	Reference
Partition	➡	Shard

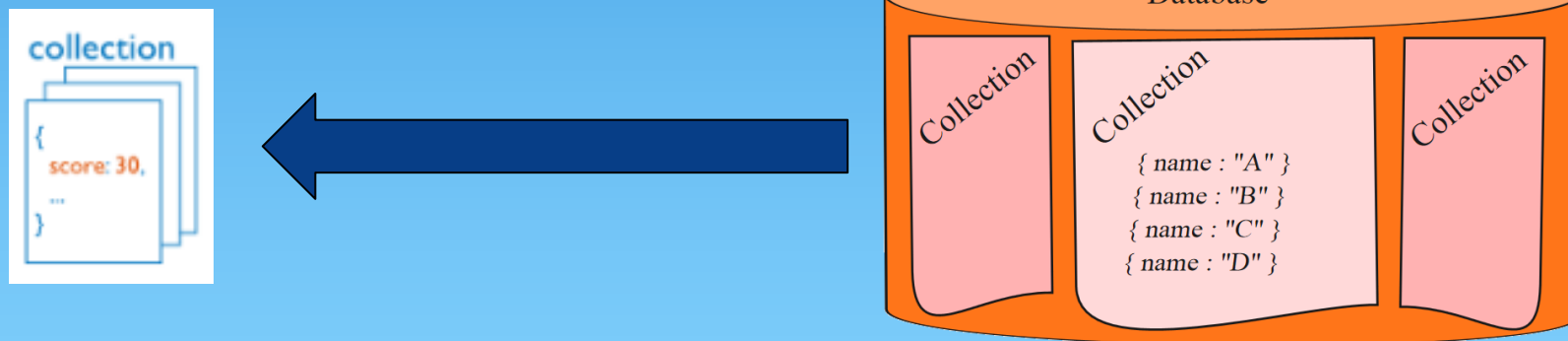




# Summary introduction

\* ב-MongoDB, מסד הנתונים מכיל אוספים (collections) של מסמכים (documents)

\* בהמשך נראה, כיצד ליצור אותם...



# BSON Example

```
> db.user.findOne({age:39})
{
  "_id" : ObjectId("5114e0bd42..."),
  "first" : "John",
  "last" : "Doe",
  "age" : 39,
  "interests" : [
    "Reading",
    "Mountain Biking ]
  "favorites": {
    "color": "Blue",
    "sport": "Soccer"}
}
```

# BSON Types

Type	Number
Double	1
String	2
Object	3
Array	4
Binary data	5
Object id	7
Boolean	8
Date	9
Null	10
Regular Expression	11
JavaScript	13
Symbol	14
JavaScript (with scope)	15
32-bit integer	16
Timestamp	17
64-bit integer	18
Min key	255
Max key	127

The number can be used with the \$type operator to query by type!

<http://docs.mongodb.org/manual/reference/bson-types/>

# The \_id Field

\* כבירת מחדל, כל Document מכיל שדה \_id לשדה הזה ישנם מספר מאפיינים מיוחדים:

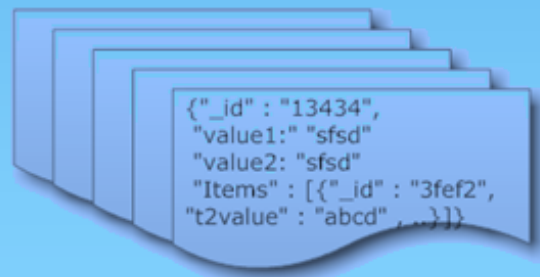
\* הערך משמש כמפתח הראשי (primary key) לאוסף (collection)

\* הערך הוא ייחודי ולא ניתן לשינוי

\* סוג ברירת המחדל הוא ObjectId שהוא "קטן, ייחודי, מהיר ליצירה וממוספר"

## Document Model

Collection ("Things")



# Getting Started with mongoDB

\* התקנה מהלינק –

<https://www.mongodb.org/downloads#production>

\* הדרכה להתקנה, נמצאת בלינק –

<https://docs.mongodb.org/manual/installation/>

(למערכות הפעלה וינדוס ולינוקס)

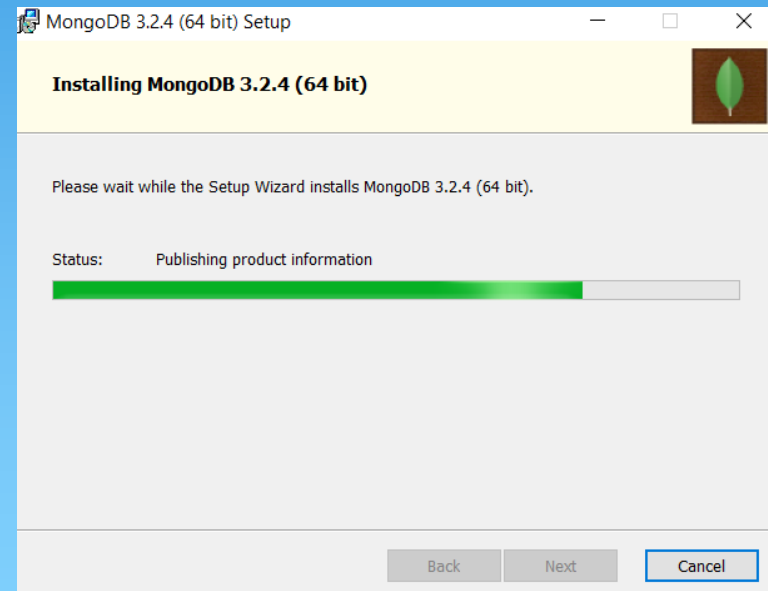
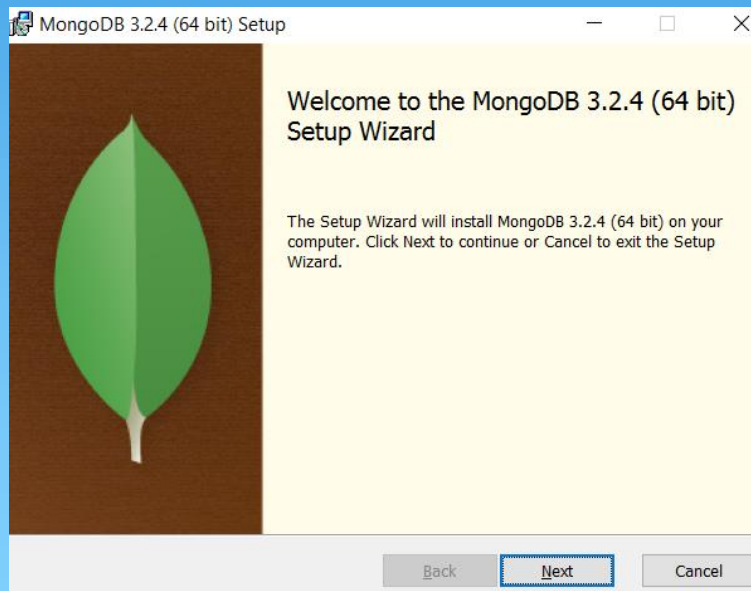


# Install mongoDB for windows

1. הורדת קובץ ההתקנה (MongoDB.msi)

2. הרצה של הקובץ, ההתקנה תיצור תיקיה בכונן C, בשם mongoDB

שימו לב: אם לא תחת c, היא נמצאת ב-Program Files ונעביר אותה ל-C-



# Run MongoDB Community Edition

## 1. הגדרת סביבת MongoDB

\* MongoDB דורש ספרייה לאחסון הנתונים

\* כברירת מחדל הנתוב לתיקייה זאת הוא: `\data\db`

\* וספרייה ללוגים: `\log`

\* ניצור את התיקיות האלו תחת הנתוב: `c:\mongodb\server\X.X`



# Run MongoDB Community Edition

2. ניצור קובץ קונפיגורציה אותו נריץ בכל פעם שנרצה להפעיל את local MongoDB server, ישנן שתי אפשרויות:

## אפשרות ראשונה:

\* ניתן ליצור קובץ config (למשל: mongo.config) ובו לכתוב את הפקודות הבאות:

**dbpath= C:\MongoDB\Server\3.2\data\db**

**logpath= C:\MongoDB\Server\3.2\log\mongodb.log**

\* בתיקייה bin נפתח חלון cmd או ב- Git Bash ונריץ את הפקודה הבאה:

**mongod.exe --config="C:\MongoDB\Server\X.X\mongo.config"**





# Run MongoDB Community Edition

## אפשרות שנייה:

\* בתיקייה bin נפתח חלון cmd או git bash ונריץ את הפקודה הבאה:

```
echo 'mongod --directoryperdb --dbpath C:/MongoDB/Server/X.X/data/db  
--logpath C:/MongoDB/Server/X.X/log/mongodb.log --logappend' > mongo-start
```

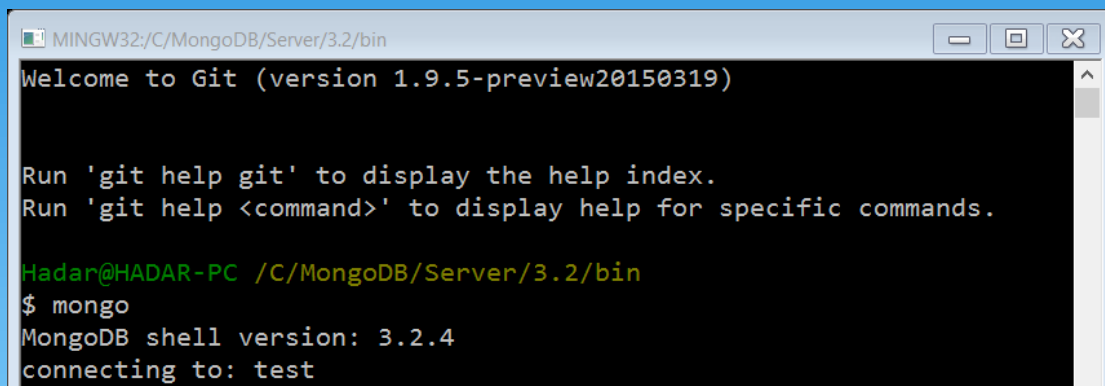
\* ואח"כ, נריץ את הפקודה:

```
chmod a-x mongo-start
```



# Test if MongoDB works

נריץ את **mongod.exe** מהתיקייה bin להפעיל את שרת ה-db המקומי  
בחלון חדש, נריץ את הפקודה mongo, על מנת לוודא החיבור:



```
MINGW32/C:/MongoDB/Server/3.2/bin
Welcome to Git (version 1.9.5-preview20150319)

Run 'git help git' to display the help index.
Run 'git help <command>' to display help for specific commands.

Hadar@HADAR-PC /C:/MongoDB/Server/3.2/bin
$ mongo
MongoDB shell version: 3.2.4
connecting to: test
```

נריץ את הפקודה db שתציג לנו את ה-database הקיימים, נוודא שנקבל  
test, test database דיפולטיבי שנוצר ע"י MongoDB כמו כן, ניתן  
להריץ את הפקודה show dbs המציגה את רשימת ה-database  
הזמינים.

# MongoDB

הסבר על קבצי ה-`.exe`:

\* `mongod` - התהליך המרכזי של בסיס הנתונים

\* `mongo` - ה-Admin Shell דרכו ניתן לבצע פעולות רבות

\* `mongoexport` - פעולות import/export של קבצי נתונים (BSON)

\* `mongorestore` / `mongodump` - גיבוי / אחזור של בסיס הנתונים

\* `mongostat` - תהליך שאוסף נתונים על השימוש ב-mongo

בגדול, כדי להפעיל ולנסות את mongo בצורה קלה יש פשוט להפעיל את `mongod.exe` ואחריו את `mongo.exe` דרך ה-console ניתן לבצע / לנסות את רוב הפעולות המשמעותיות של בסיס הנתונים – ולקבל פידבק מהיר.

# MongoDB - Using the Shell

**To check which db you're using**

**db**

**Show all databases**

**show dbs**

**Switch db's/make a new one**

**use <name>**

**See what collections exist**

**show collections**

**Note: db's are not actually created until you insert data!**



# Install mongoDB for nodejs

**Install npm install mongodb**

**Code:**

```
var MongoClient = require('mongodb').MongoClient;
var assert = require('assert'); //unit tests
var url = 'mongodb://localhost:27017/test';
MongoClient.connect(url, function(err, db) {
  assert.equal(null, err); //assert unit test
  console.log("Connected correctly to server.");
  db.close();
});
```

**Continue in the docs: <https://docs.mongodb.org/getting-started/node/client/>**



# שיטת עבודה

- \* **Install mongo**

- \* **How To Use Mongo?**

- \* You can use the shell
- \* You can use in the code
- \* You can use with additional graphical apps

- \* **How to do it right?**

- \* Test your idea in the shell on local Db
  - \* Works? Right it in the code!
- \* Have a bug in the code?
  - \* Test the code in the shell to see why the bugs keep coming

# Now and Then..

\* כאשר עובדים עם SQL, אנחנו מוגבלים לפי טבלה.

\* לכל טבלה יש מבנה משלה

\* כל נתון שנכניס חייב לעמוד בתנאי

\* לכן המון פעמים אנחנו נתקלים במצבים בהם אנחנו יוצרים יותר מטבלה אחת על מנת לייצג אוסף נתונים אחד

# תרגיל כיתה 1 - MongoDB





# SQL vs Mongo

Mongo	SQL	פעולה
אובייקט	שורה בטבלה	שמירת נתון
אוסף אובייקטים	טבלה	שמירת אוסף נתונים
Find()	ביצוע פעולה מתמטית על הטבלה	חיפוש נתון באוסף
יוצרים אובייקט	קודם צריך טבלה	יצירת נתון חדש
כל אובייקט באוסף יכול להיות עם המבנה שמתאים לו	על כל הנתונים בטבלה להיות באותו מבנה	הגבלות באוסף נתונים

# Operations in Mongo

Operation	Code
Create Collection (table in SQL) “users”	<code>db.createCollection(“users”,&lt;options&gt;)</code>
Create data	Well, just create an object
Add data to collection	<code>db.users.insert(&lt;data&gt;)</code>
Find data in collection	<code>db.users.find(&lt;data in object&gt;)</code>
Get all data from a collection	<code>db.users.find()</code>
data object type	BSON
What a data can have inside	Numbers, strings, dates, lists, object, and much more

# Operations in Mongo

Operation	Code
Update data in collection	<pre>db.users.update(   &lt;query param&gt;,   {&lt;update param&gt;:&lt;data&gt;},   &lt;options&gt; )</pre>
Update parameters	<a href="https://docs.mongodb.org/manual/reference/operator/update/#id1">https://docs.mongodb.org/manual/reference/operator/update/#id1</a>
Remove Document from a collection	<pre>db.users.remove({name:value})</pre>

# Update Parameters

Syntax	Operation
<code>\$set:{name:value}</code>	Set name to be value if found
<code>\$inc:{name:step}</code>	Increment name by step
<code>\$unset:{name:value}</code>	Remove the name (value doesn't matter)
<code>\$rename:{name:value}</code>	Change the key-name from name to value
<code>\$set:{name.\$:value}</code>	When value is as list, Will change the key matches to the query parameter inside value
<code>\$set:{name.innerNamw:value}</code>	When value is an object, Will change the inner key inside value
<code>\$pop:{name:value}</code>	Remove the value from the name list
<code>\$push:{name:value}</code>	Add the value to the name list

More on update parameters:

<https://docs.mongodb.org/manual/reference/operator/update/#id1>

# CRUD: Inserting Data

\* להכנסת נתונים ל-collection או יצירת collection חדש

```
db.<collection>.insert(<document>)
```

```
INSERT INTO <table>  
VALUES(<attributevalues>);
```

\* הוספת document עם ערך ונתון ל-collection, כ-BSON מודל

```
db.<collection>.insert({<field>:<value>})
```

\* להוספת מסמכים רבים, השתמשו במערך

# CRUD: Querying

- \* Done on collections.
- \* Get all docs: `db.<collection>.find()`
  - \* Returns a cursor, which is iterated over shell to display first 20 results.
  - \* Add `.limit(<number>)` to limit results
  - \* `SELECT * FROM <table>;`
- \* Get one doc: `db.<collection>.findOne()`

# CRUD: Querying

**\* To match a specific value:**

**\* `db.<collection>.find({ <field>:<value> })` “AND”  
`db.<collection>.find({ <field1>:<value1>, <field2>:<value2> })`**

**\* `SELECT *`  
`FROM <table>`  
`WHERE <field1> = <value1> AND <field2> = <value2>;`**

# CRUD: Querying

## \* OR

```
* db.<collection>.find({ $or: [<field>:<value1><field>:<value2>]})
```

```
* SELECT *  
FROM <table>  
WHERE <field> = <value1> OR <field> = <value2>;
```

## \* Checking for multiple values of same field

```
* db.<collection>.find({ <field>: { $in [<value>, <value>] } })
```



# CRUD: Querying

- \* Including/excluding document fields

- \* `db.<collection>.find({ <field1>:<value> }, { <field2>: 0 })`

- \* **SELECT field1  
FROM <table>;**

- \* `db.<collection>.find({ <field>:<value> }, { <field2>: 1 })`

- \* Find documents with or w/o field

- \* `db.<collection>.find({ <field>: { $exists: true } })`

# CRUD: Updating

\* **db.<collection>.update({ <field1>:<value1> },** //all docs in which field = value  
**{ \$set: { <field2>:<value2> } },** //set field to value  
**{ multi:true } )** //update multiple docs

\* **upsert:** if true, creates a new doc when none matches search criteria.

\* **UPDATE <table>**  
**SET <field2> = <value2>**  
**WHERE <field1> = <value1>;**

# CRUD: Updating

## \* To remove a field

```
* db.<collection>.update({ <field>:<value> }, { $unset: { <field>:1 } })
```

## \* Replace all field-value pairs

```
* db.<collection>.update({ <field>:<value> }, { <field>:<value>, <field>:<value> })
```

\* **NOTE:** This overwrites ALL the contents of a document, even removing fields.

# CRUD: Removal

- \* Remove all records where field = value

- \* `db.<collection>.remove({ <field>:<value> })`

- \* **DELETE FROM <table>  
WHERE <field> = <value>;**

- \* As above, but only remove first document

- \* `db.<collection>.remove({ <field>:<value> }, true)`

# CRUD

## \* Create

- \* `db.collection.insert( <document> )`
- \* `db.collection.save( <document> )`
- \* `db.collection.update( <query>, <update>, { upsert: true } )`

## \* Read

- \* `db.collection.find( <query>, <projection> )`
- \* `db.collection.findOne( <query>, <projection> )`

## \* Update

- \* `db.collection.update( <query>, <update>, <options> )`

## \* Delete

- \* `db.collection.remove( <query>, <justOne> )`



# CRUD example

```
> db.user.insert({  
  first: "John",  
  last : "Doe",  
  age: 39  
})
```

```
> db.user.find ()  
{  
  "_id" : ObjectId("51..."),  
  "first" : "John",  
  "last" : "Doe",  
  "age" : 39  
}
```

```
> db.user.update(  
  {"_id" : ObjectId("51...")},  
  {  
    $set: {  
      age: 40,  
      salary: 7000}  
    }  
  )
```

```
> db.user.remove({  
  "first": /^J/  
})
```

# תרגיל כיתה 2 - MongoDB

# MEAN Stack

