



جامعة طرابلس

كلية تقنية المعلومات

قواعد البيانات النقالة والغير متجانسة

Heterogeneous and Mobile Databases
ITMC322

أستاذ المادة / محمد أوهيبة

المحاضرة الخامسة

Review of topics

- *The heart of MongoDB – the document*
- *Understanding how MongoDB stores data*
- *Data types accepted in documents*
- *Installing and starting MongoDB*
 - *MongoDB start up options*
- *Introduction to the MongoDB shell*
 - *Inserting documents*
 - *Querying documents*
 - *Choosing the keys to return*

The heart of MongoDB – the document

- The document, an ordered set of keys with associated values.
- The representation of a document varies by the programming language, but most languages have a data structure that is a natural fit, such as a map, hash, or dictionary. Here is a very basic example of a document, which is understood by MongoDB:

```
{ "name" : "Ahmed",  
  "age" : 44,  
  "phone" : "092-567-890" }
```

- Most documents will be more complex than this simple one and will often contain embedded data within them. These denormalized data models allow applications to retrieve and manipulate related data in a single database operation:

```
{ "name" : " Ahmed ",  
  "age" : 44,  
  "contact" : {  
    "phone" : " 092-567-890"  
  }  
}
```

- we have included the contact information within the same document by using an embedded document with a single key named `contact`.

The heart of MongoDB the document (Cont.)

- Each document requires a key, which needs to be unique within a document.
- The keys contained in a document are strings. Any UTF-8 character can be included in a key, with a few exceptions:
- You cannot include the character `\0` (also known as the null character) in a key. This character is used to indicate the end of a key.
- The `.` and `$` characters are internally used by the database so they should be used only in limited cases. As a general rule, it is better to completely avoid using these characters as most MongoDB drivers can generate exceptions when they are used inappropriately.
- MongoDB is both type-sensitive and case-sensitive. For example, these documents are distinct:
 - `{ "age" : 18 }`
 - `{ "age" : "18" }`
- The same applies to the following documents:
 - `{ "age" : 18 }`
 - `{ "Age" : 18 }`

Understanding how MongoDB stores data

- **JavaScript Object Notation** (JSON) is a standard that simplifies data interchange in applications
- JSON is able to deal with all the data types: numbers, Boolean values, arrays, and objects
- MongoDB is able to store JSON documents. See an example of a JSON document below.
- A JSON-based database returns data in a format that programming languages such as JavaScript require a small amount of code you need to load

```
{
  "_id":1,
  "name":{
    "first":"Dennis",
    "last":"Ritchie"
  },
  "contribs":[
    "Altran",
    "B",
    "C",
    "Unix"
  ],
  "awards":[
    {
      "award":"Turing Award",
      "year":1983
    },
    {
      "award":"National medal of technology",
      "year":1999
    }
  ]
}
```

Understanding how MongoDB stores data (Cont.)

- MongoDB represents JSON documents using a binary-encoded format called **BSON**.
- MongoDB stores documents (objects) in a format called BSON. BSON is a binary serialization of JSON-like documents. BSON stands for “Binary JSON”, but also contains extensions that allow representation of data types that are not part of JSON. For example, BSON has a Date data type and BinData type.
- Documents encoded with BSON enhance the JSON data model to provide additional data types and efficiency when encoding/decoding data within different languages.
- MongoDB uses a fast and lightweight BSON implementation, which is highly traversable and supports complex structures such as embedded objects and arrays.

Understanding how MongoDB stores data (Cont.)

BSON was designed to have the following three characteristics:

1. Lightweight

Keeping spatial overhead to a minimum is important for any data representation format, especially when used over the network.

2. Traversable

BSON is designed to be traversed easily. This is a vital property in its role as the primary data representation for MongoDB.

3. Efficient

Encoding data to BSON and decoding from BSON can be performed very quickly in most languages due to the use of C data types.

Data types accepted in documents

- MongoDB offers a wide choice of data types, which can be used in your documents:
 - **String:** This is the most common data type as it contains a string of text (such as: "name" : "John").
 - **Integer (32 bit and 64-bit):** This type is used to store a numerical value (for example, "age" : 40).
- Note that an Integer requires no quotes to be placed before or after the Integer.
- **Boolean:** This data type can be used to store either a `TRUE` or a `FALSE` value.
 - **Double:** This data type is used to store floating-point values.
 - **Min/Max keys:** This data type is used to compare a value against the lowest and highest BSON elements, respectively.
 - **Arrays:** This type is used to store arrays or list or multiple values into one key (for example, ["John, Smith", "Mark, Spencer"]).
 - **Timestamp:** This data type is used to store a timestamp. This can be useful to store when a document has been last modified or created.
 - **Object:** This data type is used for storing embedded documents.
 - **Null:** This data type is used for a null value.
 - **Symbol:** This data type allows storing characters such as String; however, it's generally used by languages that use a specific symbol type.
 - **Date:** This data type allows storing the current date or time in the Unix time format (POSIX time).
 - **Object ID:** This data type is used to store the document's ID.
 - **Binary data:** This data type is used to store a binary set of data.
 - **Regular expression:** This data type is used for regular expressions. All options are represented by specific characters provided in alphabetical order. we will learn more about regular expressions.
 - **JavaScript code:** This data type is used for JavaScript code.

Installing and starting MongoDB

- Installing Mongo DB is much easier than most RDBMS as it's just a matter of unzipping the archived database and, if necessary, configure a new path for data storage.
- **Installing MongoDB on Windows**
- For installing MongoDB on Windows, perform the following steps:
 - Download the latest stable release of MongoDB from <http://www.mongodb.org/downloads>. (At the time of writing, the latest stable release is 3.0.3, which is available as Microsoft Installer or as a ZIP file). Ensure you download the correct version of MongoDB for your Windows system.
- Execute the MSI Installer, or if you have downloaded MongoDB as a ZIP file, simply extract the downloaded file to `C:\drive` or any other location.
- MongoDB requires a data directory to store its files. The default location for the MongoDB data folder on Windows is `c:\data\db`. Execute the following command from the command prompt to create the default folder:
`C:\mongodb-win32-x86_64-3.0.3>md data`

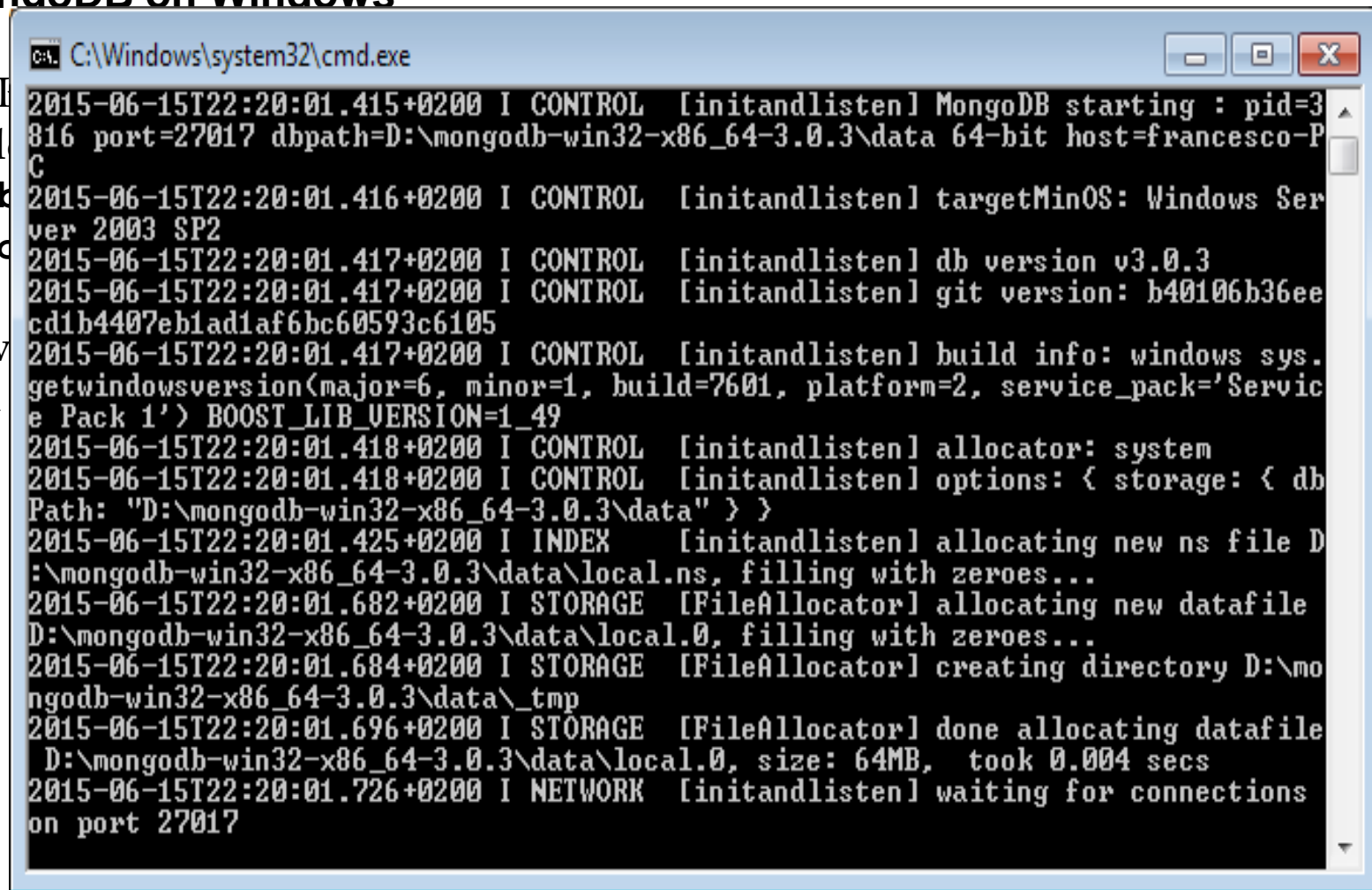
Installing and starting MongoDB

- **Installing MongoDB on Windows**

- In Command Prompt, run the installation folder

```
C:\mongodb> mongod  
"C:\mongodb>
```

- This will show the output, which



```
C:\Windows\system32\cmd.exe  
2015-06-15T22:20:01.415+0200 I CONTROL [initandlisten] MongoDB starting : pid=3816 port=27017 dbpath=D:\mongodb-win32-x86_64-3.0.3\data 64-bit host=francesco-PC  
2015-06-15T22:20:01.416+0200 I CONTROL [initandlisten] targetMinOS: Windows Server 2003 SP2  
2015-06-15T22:20:01.417+0200 I CONTROL [initandlisten] db version v3.0.3  
2015-06-15T22:20:01.417+0200 I CONTROL [initandlisten] git version: b40106b36eed1b4407eb1ad1af6bc60593c6105  
2015-06-15T22:20:01.417+0200 I CONTROL [initandlisten] build info: windows sys.getwindowsversion(major=6, minor=1, build=7601, platform=2, service_pack='Service Pack 1') BOOST_LIB_VERSION=1_49  
2015-06-15T22:20:01.418+0200 I CONTROL [initandlisten] allocator: system  
2015-06-15T22:20:01.418+0200 I CONTROL [initandlisten] options: { storage: { dbPath: "D:\mongodb-win32-x86_64-3.0.3\data" } }  
2015-06-15T22:20:01.425+0200 I INDEX [initandlisten] allocating new ns file D:\mongodb-win32-x86_64-3.0.3\data\local.ns, filling with zeroes...  
2015-06-15T22:20:01.682+0200 I STORAGE [FileAllocator] allocating new datafile D:\mongodb-win32-x86_64-3.0.3\data\local.0, filling with zeroes...  
2015-06-15T22:20:01.684+0200 I STORAGE [FileAllocator] creating directory D:\mongodb-win32-x86_64-3.0.3\data\_tmp  
2015-06-15T22:20:01.696+0200 I STORAGE [FileAllocator] done allocating datafile D:\mongodb-win32-x86_64-3.0.3\data\local.0, size: 64MB, took 0.004 secs  
2015-06-15T22:20:01.726+0200 I NETWORK [initandlisten] waiting for connections on port 27017
```

MongoDB start up options

- The list of start up options and is detailed at [mongod/](#).
- The following table

Option	Description
<code>--help, -h</code>	This returns the information on the options and use of mongod.
<code>--version</code>	This returns the mongod release number.
<code>--config <filename></code>	This specifies the configuration file to be used by mongod.
<code>--port <port></code>	This specifies the TCP listening port on which MongoDB listens. (the default is 27017)
<code>--bind_ip <ip address></code>	This specifies the IP address that mongod binds to in order to listen for connections from applications (the default is All interfaces.).
<code>--logpath <path></code>	This sends all diagnostic logging information to a log file instead of to a standard output or to the host's syslog system.
<code>--logappend</code>	This appends new entries to the end of the log file rather than overwriting the content of the log when the mongod instance restarts.
<code>--httpinterface</code>	This enables the HTTP interface. Enabling the interface can increase network exposure.
<code>--fork</code>	This enables a daemon mode that runs the mongod process in the background. By default, mongod does not run as a daemon.
<code>--auth</code>	This enables authorization to control the user's access to database resources and operations. When authorization is enabled, MongoDB requires all clients to authenticate themselves first in order to determine the access for the client.
<code>--noauth</code>	This disables authentication. It is currently the default and exists for future compatibility and clarity.
<code>--rest</code>	This enables the simple REST API. Enabling the REST API enables the HTTP interface, even if the HTTP interface option is disabled, and as a result can increase network exposure.

3e /

MongoDB start up options (Cont.)

- The list of start up options and is detailed at <http://mongodb/>.
- The following table su:

Option	Description
<code>--profile <level></code>	This changes the level of database profiling (0 Off, which means no profiling; 1 On, which only includes slow operations; and 2 On, which includes all the operations.)
<code>--shutdown</code>	This safely terminates the mongod process. It is available only on Linux systems.

In addition, the following options can be used to vary the storage of the database:

Option	Description
<code>--dbpath <path></code>	This is the directory where the mongod instance stores its data. The default is <code>/data/db</code> on Linux and OS X and <code>C:\data\db</code> on Windows.
<code>--storageEngine string</code>	This specifies the storage engine for the mongod database. The valid options include <code>mmapv1</code> and <code>wiredTiger</code> . The default is <code>mmapv1</code> .
<code>--directoryperdb</code>	This stores each database's files in its own folder in the data directory. When applied to an existing system, the <code>--directoryperdb</code> option alters the storage pattern of the data directory.

Mongo tools

- MongoDB ships with a set of shell commands, which can be useful to administrate your server.
- an initial introduction to the server administration:
- `bsondump`: This displays BSON files in a human-readable format
- `mongoimport`: This converts data from JSON, TSV, or CSV and stores them into a collection
- `mongoexport`: This writes an existing collection using the CSV or JSON formats
- `mongodump/mongorestore`: This dumps MongoDB data to disk using the BSON format (`mongodump`), or restores them (`mongorestore`) to a live database
- `mongostat`: This monitors running MongoDB servers, replica sets, or clusters
- `mongofiles`: This reads, writes, deletes, or updates files in GridFS
- `mongooplog`: This replays oplog entries between MongoDB servers
- `mongotop`: This monitors data reading/writing on a running Mongo server
- **an example** of how to use the `mongoimport` tool to import a CSV-formatted data contained in `/var/data/users.csv` into the collection `users` in the sample database on the MongoDB instance running on the localhost port numbered 27017:
- **`mongoimport --db sample --collection users --type csv --headerline --file /var/data/users.csv`**

Mongo tools (Cont.)

- If you want to export the MongoDB documents, you can use the `mongoexport` tool. Let's look at an example of how to export the collection `users` (part of the sampled database), limited to the first 100 records:
- **`mongoexport --db sampledb --collection users --limit 100 --out export. Json`**
- As part of your daily backup strategy, you should consider using the `mongodump` tool, which is a utility for creating a binary export of the contents of a database.
- The following command creates a database dump for the collection named `users` contained in the database named `sampled`. In this case, the database is running on the local interface on port 27017:
`mongodump --db test --collection users`
- The preceding command will create a BSON binary file named `users.bson` and a JSON file named `users.metadata.json` containing the documents. The files will be created under `dump/[database-name]`.

Introduction to the MongoDB shell

- MongoDB ships with a JavaScript shell that allows interaction with a MongoDB instance from the command line. The shell is the bread-and-butter tool for performing administrative functions, monitoring a running instance, or just inserting documents.
- To start the shell, run the `mongo` executable:

```
$ mongo
MongoDB shell version: 3.0.3
connecting to: test
```
- The shell automatically attempts to connect to a running MongoDB server on startup, so make sure you start `mongod` before starting the shell.
- If no other database is specified on startup, the shell selects a default database called `test`.
- let's start by switching to the `sampledb` database:

```
> use sampledb
switched to db sampledb
```
- If you want to check the list of available databases, then you can use the `show dbs` command:

```
>show dbs
local 0.78125GB
test 0.23012GB
```

Introduction to the MongoDB shell (Cont.)

- **Inserting documents**
- MongoDB documents can be specified in the JSON format. For example, let's recall the simple document that we have already introduced:

```
{ "name" : "Ahmed",  
  "age" : 44,  
  "phone": "092-567-890"  
}
```

- In order to insert this document, you need to choose a collection where the document will be stored. Here's how you can do it with the Mongo shell:

```
db.users.insert( {"name": "Ahmed", "age": 44, "phone": "092-567-  
890" } )
```

- As for databases, collections can be created dynamically by specifying it into the `insert` statement.

Introduction to the MongoDB shell (Cont..)

- **Querying documents**
- The `find` method is used to perform queries in MongoDB. If no argument is given to the `find` method, it will return all the documents contained in the collection as in the following statement:

- `> db.users.find()`

- The response will look something like this:

```
{ "_id" : ObjectId("5506d5988d7bd8471669e675"), "name" : "Ahmed",  
"age" : 44, "phone" : "092-456-789" }
```

- The `_id` field has been added to the document. This is a special key that works like a primary key.
- Every MongoDB document requires a unique identifier and if you don't provide one in your document, then a special MongoDB ID will be generated and added to the document at that time.

Introduction to the MongoDB shell (Cont...)

- **Querying documents**
- Now, let's include another user in our collections so that we can refine our searches:

```
> db.users.insert({"name": "Ali", "age": 32, "phone": "091-444-333"})
```
- Your collection should now include two documents, as verified by the **count** function:

```
> db.users.count()  
2
```
- Having two documents in our collection, we will learn how to add a query selector to our `find` statement so that we filter users based on a key value. For example, here is how to find a user whose name is `Ali`:

```
> db.users.find({"name": "Ali"})
```

```
{ "_id" : ObjectId("5506eea18d7bd8471669e676"), "name" :  
"Ali", "age" : 32, "phone" : "091-444-333" }
```

Introduction to the MongoDB shell (Cont...)

- **Querying documents**
- Multiple conditions can be specified within a query, just like you would do with a WHERE - AND construct in SQL:

```
> db.users.find({"name": "Ali", "age": 32})
```

```
{ "_id" : ObjectId("5506eea18d7bd8471669e676"), "name" :  
"Ali", "age" : 32, "phone" : "091-444-333" }
```

Introduction to the MongoDB shell (Cont....)

- **Choosing the keys to return**
- The queries mentioned earlier are equivalent to a `SELECT *` statement in SQL terms.
- You can use a projection to select a subset of fields to return from each document in a query result set.
- This can be especially useful when you are selecting large documents, as it will reduce the costs of network latency and deserialization.
- Projections are commonly activated by means of binary operators (0,1); the binary operator `0` means that the key must not be included in the search whilst `1` obviously means that the key has to be included.
- an example of how to include the `name` and `age` keys in the fields to be returned (along with the `id` field, which is always included by default:

```
> db.users.find({}, {"name": 1,"age": 1})
```

```
{ "_id" : ObjectId("5506d5988d7bd8471669e675"), "name" : "Ahmed", "age" : 44 }
```

```
{ "_id" : ObjectId("5506eea18d7bd8471669e676"), "name" : "Ali", "age" : 32 }
```

- By setting the projection values for the `name` and `age` to `0`, the phone number is returned instead:

```
> db.users.find({}, {"name": 0,"age": 0})
```

```
{ "_id" : ObjectId("5506d5988d7bd8471669e675"), "phone" : "092-456-789" }
```

```
{ "_id" : ObjectId("5506eea18d7bd8471669e676"), "phone" : "091-444-333" }
```

Reference Book

- **MongoDB for Java Developers_ Design, build, and deliver efficient Java applications using the most advanced NoSQL database.**

Francesco Marchioni

Copyright © 2015 Packt Publishing



The End

Thanks for listening ..

Any questions ?