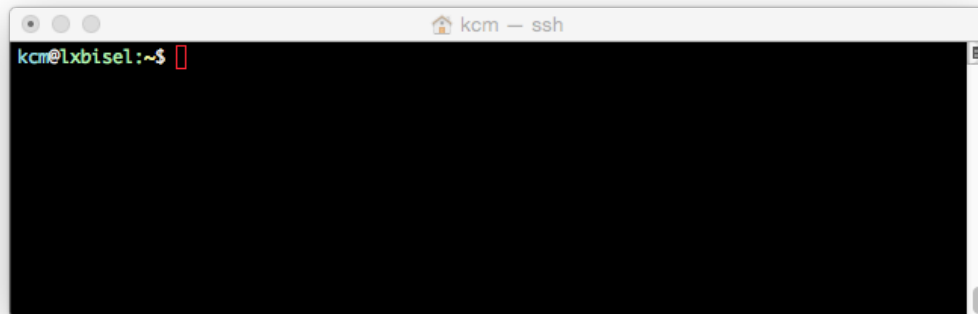


MongoDB lab

This simple tutorial will guide you through the basics of using MongoDB. You will be set a number of tasks and a final exercise. Should you have any problems, please ask for help.

CONNECTING

Open a terminal (linux command line) window:



Connect to mongoDB using the mongo client:

```
mongo -u username -p --authenticationDatabase username
--host mongo-server-1
```

When prompted, enter your password

You have opened the test database, switch to your personal database:

```
use username
```

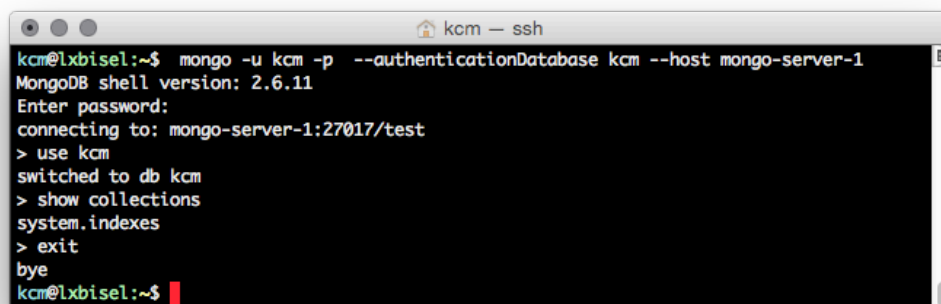
Look at the collections that currently exist (nothing should appear as your database is empty):

```
show collections
```

Now exit mongo and return to the terminal's command line:

```
exit
```

Your terminal window should look similar to:



IMPORTING DATA

Download the data for this lab from:

<http://www.macs.hw.ac.uk/~kcm/teaching/dbis/labData.json>

Open the file and look at the data. Notice that there is one JSON document per line.

Using the linux command line, insert the example data into your database using mongoimport:

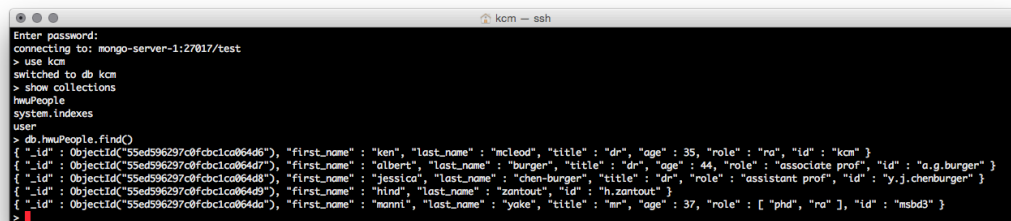
```
mongoimport --db username -u username -p password
--host mongo-server-1 --authenticationDatabase username
--collection hwuPeople < labData.json
```

Repeat the above steps (from CONNECTING) to logon and view the collections. When you look at the collections you should now see a "hwuPeople" collection.

Inspect the contents of hwuPeople:

```
db.hwuPeople.find()
```

You will see 5 people listed:



```
kcm ~ ssh
Enter password:
connecting to: mongo-server-1:27017/test
> use kcm
switched to db kcm
> show collections
hwuPeople
system.indexes
user
> db.hwuPeople.find()
{ "_id" : ObjectId("55ed596297c0fbc1ca064d8"), "first_name" : "ken", "last_name" : "mcLeod", "title" : "dr", "age" : 35, "role" : "ra", "id" : "kcm" }
{ "_id" : ObjectId("55ed596297c0fbc1ca064d7"), "first_name" : "albert", "last_name" : "burger", "title" : "dr", "age" : 44, "role" : "associate prof", "id" : "a.g.burger" }
{ "_id" : ObjectId("55ed596297c0fbc1ca064d8"), "first_name" : "jessica", "last_name" : "chen-burger", "title" : "dr", "role" : "assistant prof", "id" : "y.j.chenburger" }
{ "_id" : ObjectId("55ed596297c0fbc1ca064d9"), "first_name" : "hind", "last_name" : "zantout", "id" : "h.zantout" }
{ "_id" : ObjectId("55ed596297c0fbc1ca064da"), "first_name" : "manni", "last_name" : "yake", "title" : "mr", "age" : 37, "role" : [ "phd", "ra" ], "id" : "msbd3" }
```

In each person you will see something like:

```
{ "_id" : ObjectId("541ff6239c494c635a2d32d7"), "first_name" : ...
```

This "_id" is the internal ID that is used to uniquely identify documents. It is the only part of the document that is indexed by default. I.e., this is the primary key.

QUERYING

Let us find all the people with role "ra":

```
db.hwuPeople.find({"role" : "ra"})
```

Notice that Manni appears in the output: mongo automatically searches inside an array.

To find all people older than 35:

```
db.hwuPeople.find({age : {$gt: 35}})
```

There are multiple ways to write the same query, e.g.:

```
db.hwuPeople.aggregate([{$match : { age : {$gt: 35}}}])
```

TASK: write a query to find all the RAs under 40 years old. Hint: use \$lt for less than.

Normally, when multiple conditions are specified they use AND logic. However, you can specify OR instead; find all people that have a role "assistant prof" or "associate prof":

```
db.hwuPeople.find({$or: [{role: 'associate prof'}, {role: 'assistant prof'}]})
```

To find everyone who does not have an "age" specified:

```
db.hwuPeople.find({age : {$exists: false}})
```

Rather than listing the documents without an age, we can simply ask for the number of documents:

```
db.hwuPeople.count({age : {$exists: false}})
```

To sort the results of a "find" query append `.sort({name: 1})` after the `find()` statement. Use 1 for ascending order, and -1 for descending order.

TASK: Sort the entire list of people in ascending order by age.

INSERTING, UPDATING & REMOVING

To insert someone into the database:

```
db.hwuPeople.insert({first_name : "joe", last_name : "blogs", age : 21, role : "msc", id : "jb33"})
```

To prove this worked:

```
db.hwuPeople.find({first_name : "joe"})
```

TASK: add yourself, specifying your `_id` manually. HINT: treat "`_id`" as just another name/value pair.

Dr Burger has been promoted and so we need to change his "title" and his "role" to "prof":

```
db.hwuPeople.update({last_name : "burger"},{$set: {title : "prof", role : "prof"}})
```

The basic form of an update query is:

```
({query}, {change})
```

When a document matches the "query" it is altered to reflect the "change".

With a flexible schema you can add information to one document but not to others: add an email address for Dr M^cLeod:

```
db.hwuPeople.update({last_name: "mcleod"}, {$set: {email: "kcm1@hw.ac.uk"}})
```

Look at the document for Manni:

```
db.hwuPeople.find({first_name : "manni"})
```

Manni is listed as being 37, but his birthday was last week; to increase his age:

```
db.hwuPeople.update({first_name : "manni"}, {$inc: {age: 1}})
```

Manni has another role (lab assistant), to add this:

```
db.hwuPeople.update({first_name: "manni"}, {$push: {role: "lab assistant"}})
```

NOTE: `$push` only works for arrays.

TASK: update your information to provide your email address and your title (e.g., mr, ms etc.).

If you try to update a document that is not there, nothing happens:

```
db.hwuPeople.update({first_name: "andy", last_name: "proudlove",
  role: "ra"}, {age: 47})

db.hwuPeople.find()
```

MongoDB supports "upserts"; this form of query will try to update an existing document, but if it does not exist it will insert it. However, you have to be careful with upserts, try the following:

```
db.hwuPeople.update({first_name: "andy", last_name: "proudlove",
  role: "ra"}, {age: 47}, {upsert:true})
db.hwuPeople.find()
```

Notice what has happened: a new document has been added, but it only contains "age". Clearly, this is wrong! To remove this document (*careful*: if your age is also 47 it will delete you too):

```
db.hwuPeople.remove({age: 47})
```

TASK: Write an upsert query to properly insert Andy Proudlove into the database.

Hint: The basic form of this query will be:

```
{query}, {change}, {upsert:true}
```

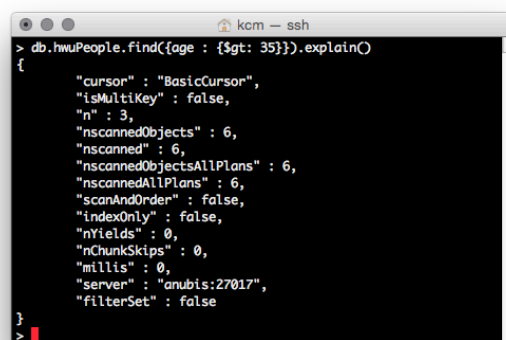
TASK: Use the *remove* command to delete Joe Bloggs from the database. NOTE: most of the operations that work with the *insert* command also work with the *remove* command.

OPTIMISATION

For queries to be efficient they must use an index. To check if a query uses an index use the `.explain()` method:

```
db.hwuPeople.find({age : {$gt: 35}}).explain()
```

We can see that the cursor used is the "BasicCursor", which means no index was used:



```
kcm - ssh
> db.hwuPeople.find({age : {$gt: 35}}).explain()
{
  "cursor" : "BasicCursor",
  "isMultiKey" : false,
  "n" : 3,
  "nscannedObjects" : 6,
  "nscanned" : 6,
  "nscannedObjectsAllPlans" : 6,
  "nscannedAllPlans" : 6,
  "scanAndOrder" : false,
  "indexOnly" : false,
  "nYields" : 0,
  "nChunkSkips" : 0,
  "millis" : 0,
  "server" : "anubis:27017",
  "filterSet" : false
}
```

To add an index to "age":

```
db.hwuPeople.ensureIndex({age: 1})
```

Now if we run the `.explain()` method again we see the cursor is "BtreeCursor". This is an efficient index, based on B-trees¹, and so our query is optimized.

BACKUP

Exit mongo

Then use `mongoexport` to backup your current database:

```
mongodump -u username -p password --host mongo-server-1
--authenticationDatabase username -d username -c hwuPeople
```

In your home directory you should see a "dump" folder. Your files are in there. You can copy them to another machine and use `mongoimport` to import them.

DELETING A COLLECTION

Log into mongo and switch to your database.

To delete the `hwuPeople` collection:

```
db.hwuPeople.drop()
show collections
```

EXERCISE

Create a new collection (called "exercise") that includes the following information:

```
name: albert burger, role: supervisor
name: alasdair grey, role: supervisor
name: iain wiles, role: phd
name: steve smith, role: phd
name: hugh dollar, role: phd
```

Additionally, include the following relationships:

Alasdair supervises Iain and Steve.

Albert supervises Steve and Hugh.

HINT: use an array to hold the supervisor information.

Now construct an optimized query to list all of Alasdair Grey's students.

Write a second query to find all the students with 2 supervisors.

HINT: use the condition `$size: 2`.

¹ <http://en.wikipedia.org/wiki/B-tree>