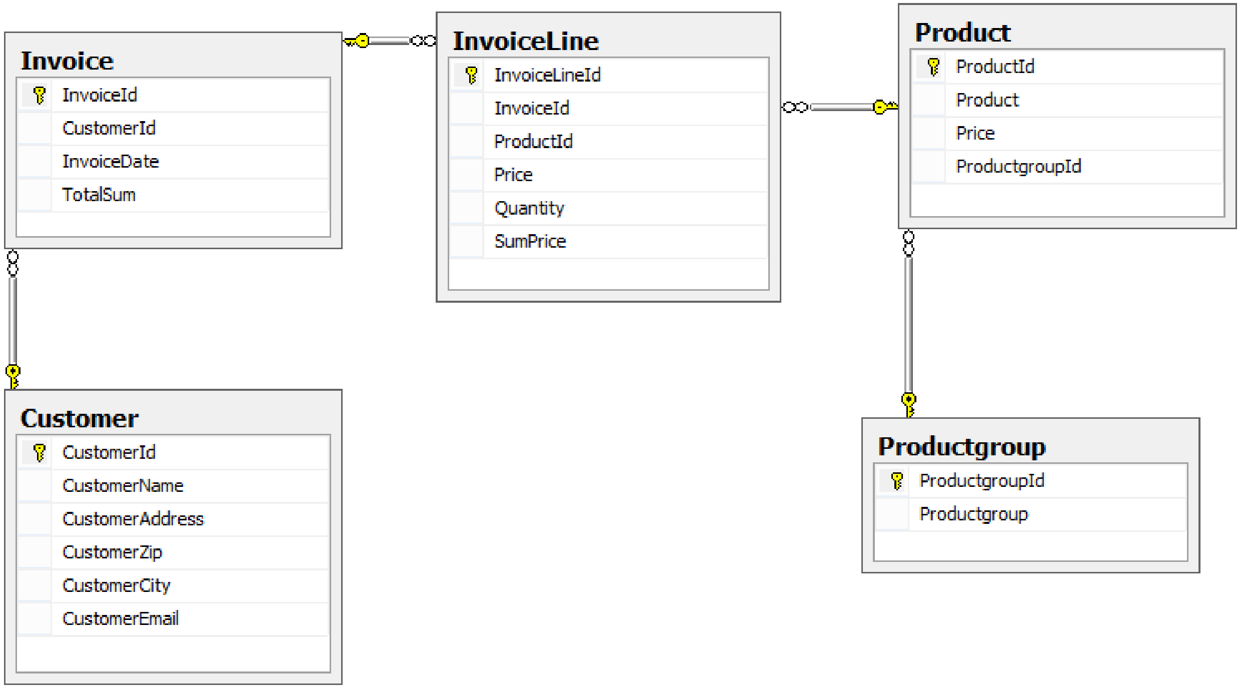
BUSINESS ACADEMY
AARHUS

Databases & XML (11) - 25.04.2016



Invoice
- InvoiceId
- CustomerId
- InvoiceDate
- TotalSum

InvoiceLine
- InvoiceLineId
- InvoiceId
- ProductId
- Price
- Quantity
- SumPrice

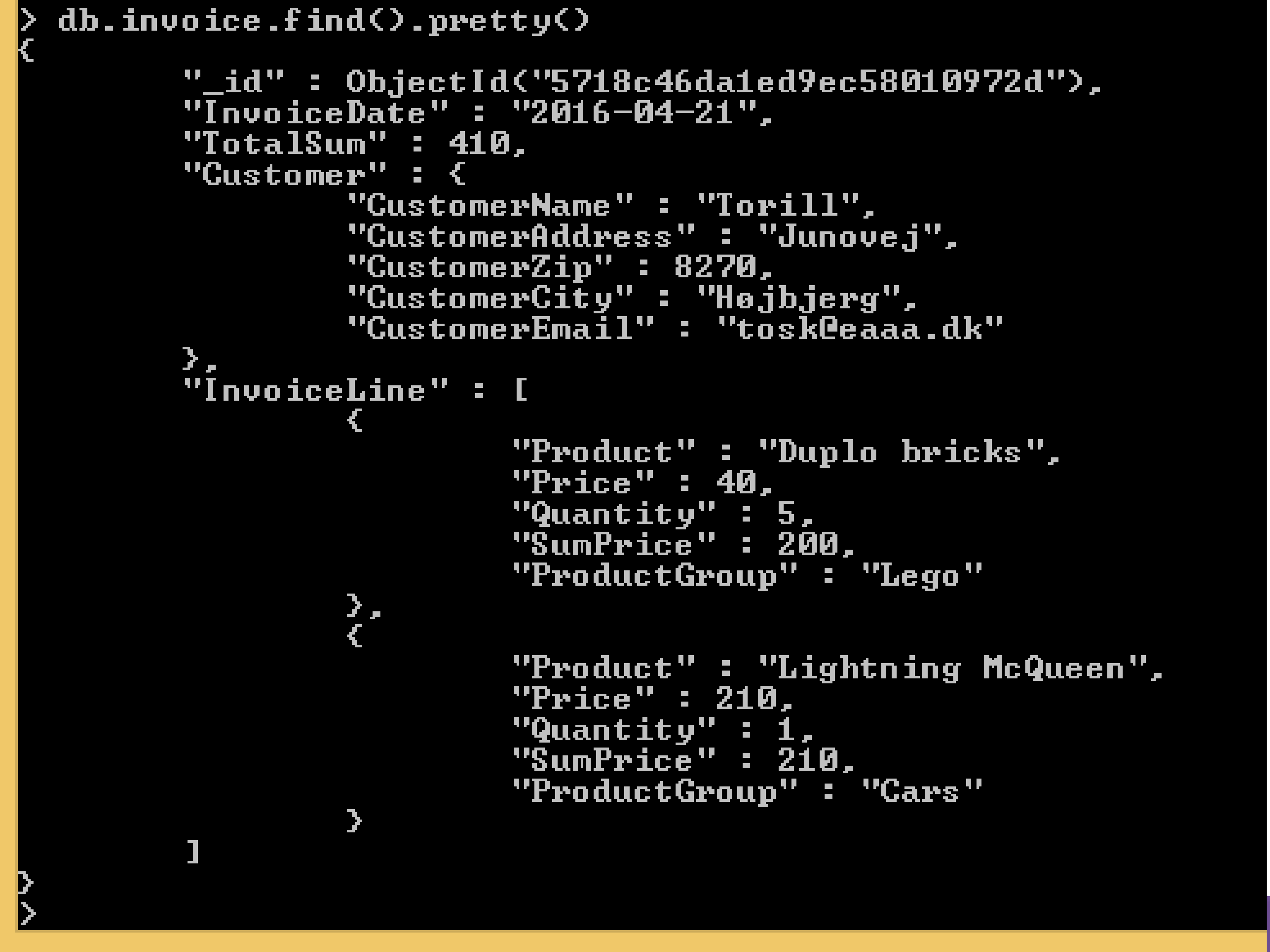
Product
- ProductId
- Product
- Price
- ProductgroupId

Customer
- CustomerId
- CustomerName
- CustomerAddress
- CustomerZip
- CustomerCity
- CustomerEmail

Productgroup
- ProductgroupId
- Productgroup

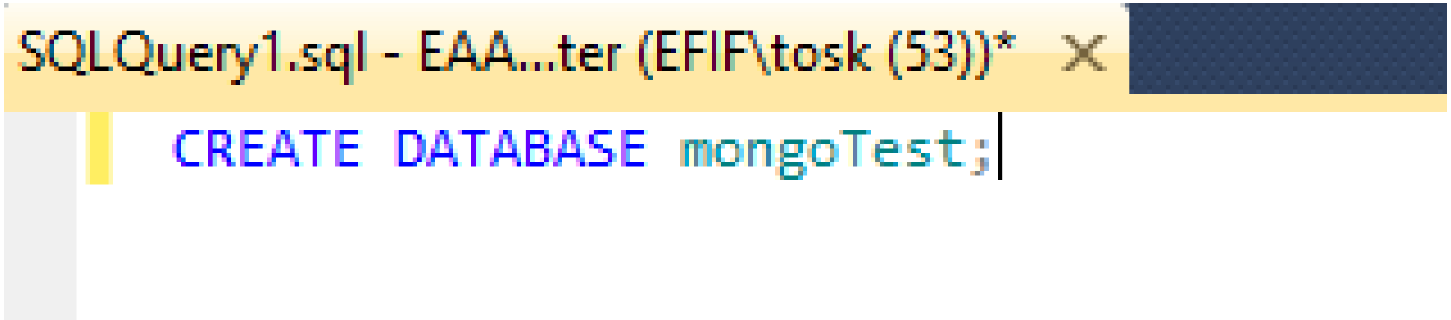
Create a database (webshop) with the relevant collection for selling products

Exercise 3



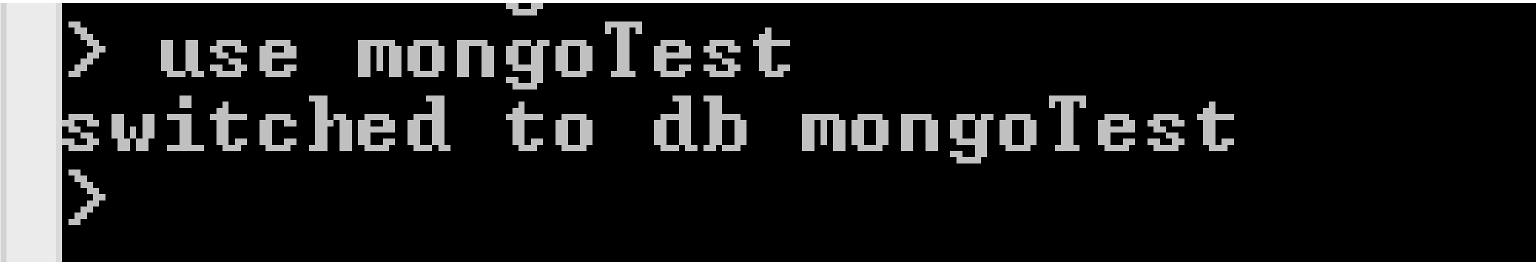
```
> db.invoice.find().pretty()
{
        "_id" : ObjectId("5718c46da1ed9ec58010972d"),
        "InvoiceDate" : "2016-04-21",
        "TotalSum" : 410,
        "Customer" : {
                "CustomerName" : "Torill",
                "CustomerAddress" : "Junovej",
                "CustomerZip" : 8270,
                "CustomerCity" : "Højbjerg",
                "CustomerEmail" : "tosk@eaaa.dk"
        },
        "InvoiceLine" : [
                {
                        "Product" : "Duplo bricks",
                        "Price" : 40,
                        "Quantity" : 5,
                        "SumPrice" : 200,
                        "ProductGroup" : "Lego"
                },
                {

                        "Product" : "Lightning McQueen",
                        "Price" : 210,
                        "Quantity" : 1,
                        "SumPrice" : 210,
                        "ProductGroup" : "Cars"
                }
        ]
}
>
```

Exercise 3 ”solution”



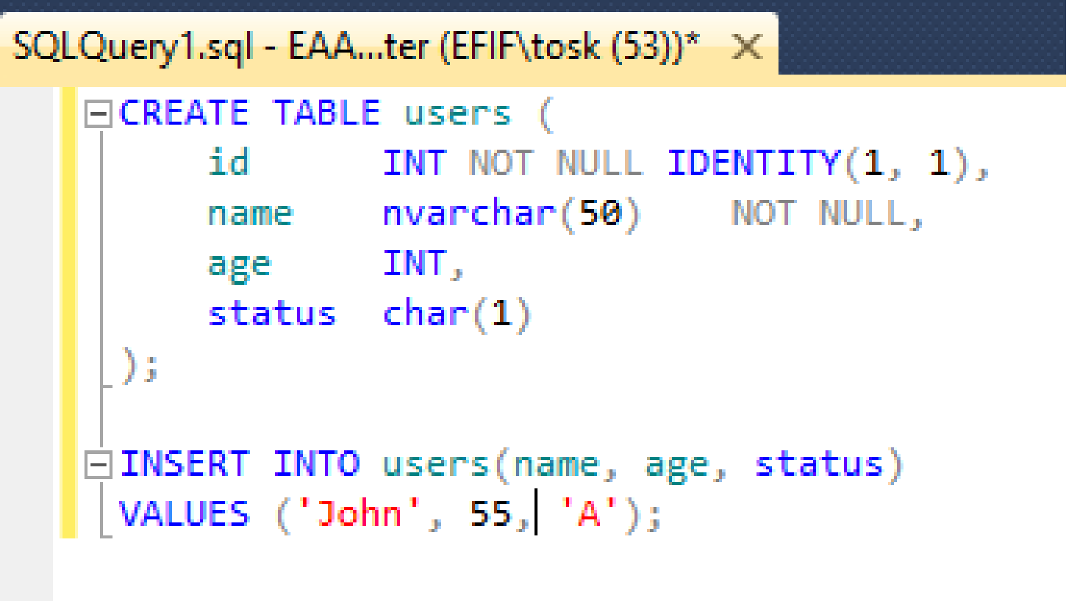
SQLQuery1.sql - EAA...ter (EFIF\tosk (53))*

```sql
CREATE DATABASE mongoTest;
```



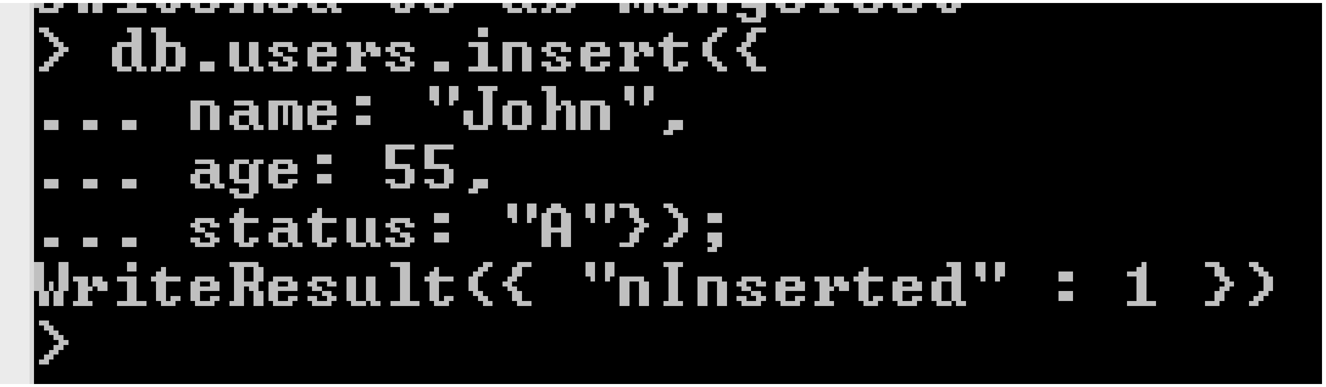
```
> use mongoTest
switched to db mongoTest
>
```

Create database



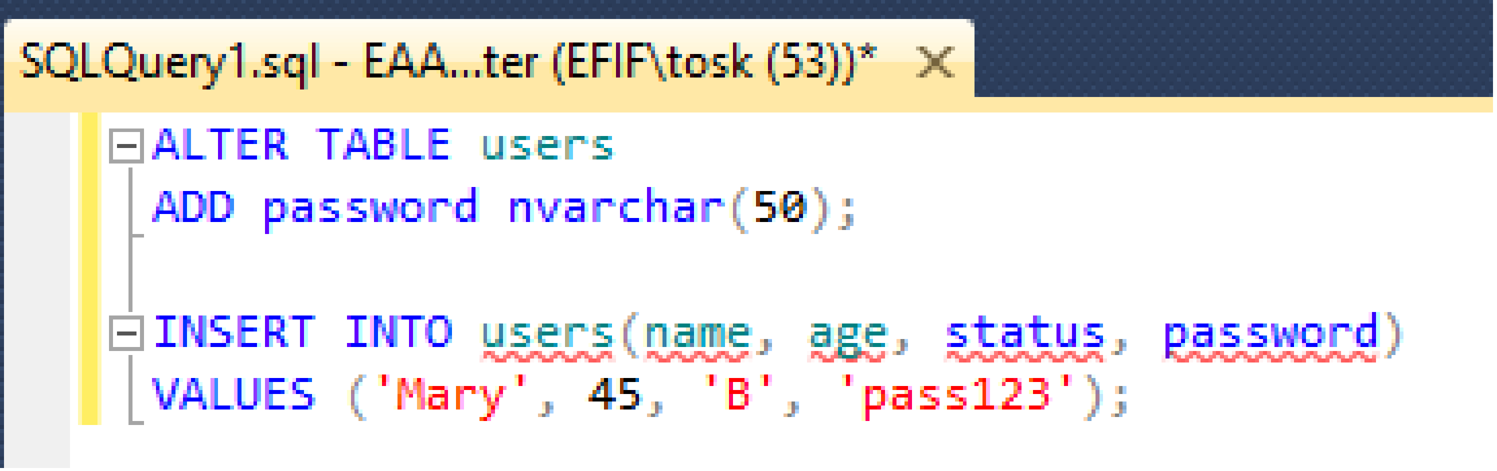
SQLQuery1.sql - EAA...ter (EFIF\tosk (53))*

```sql
CREATE TABLE users (
    id        INT NOT NULL IDENTITY(1, 1),
    name      nvarchar(50)    NOT NULL,
    age       INT,
    status    char(1)
);

INSERT INTO users(name, age, status)
VALUES ('John', 55, 'A');
```

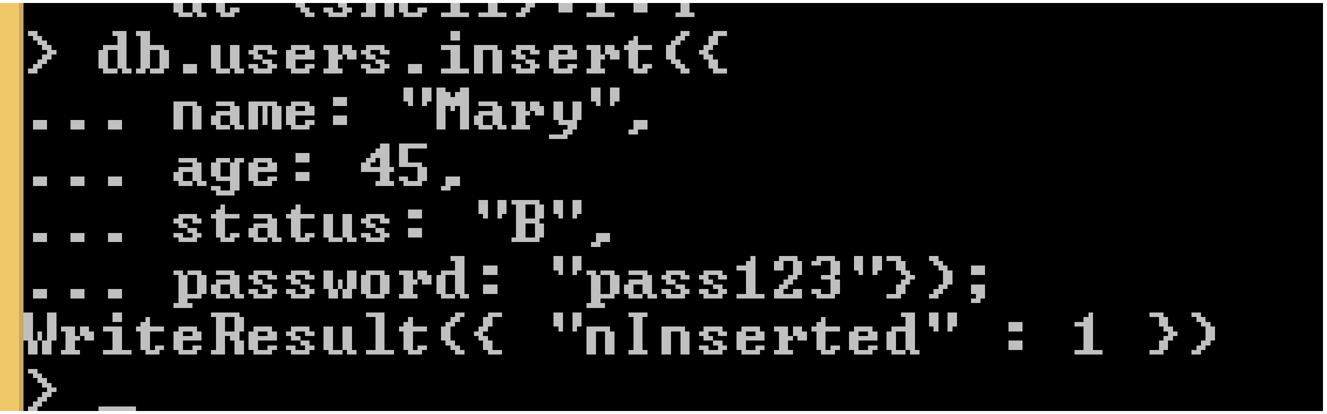


```
> db.users.insert({
... name: "John",
... age: 55,
... status: "A"});
WriteResult({ "nInserted" : 1 })
>
```

Create a user

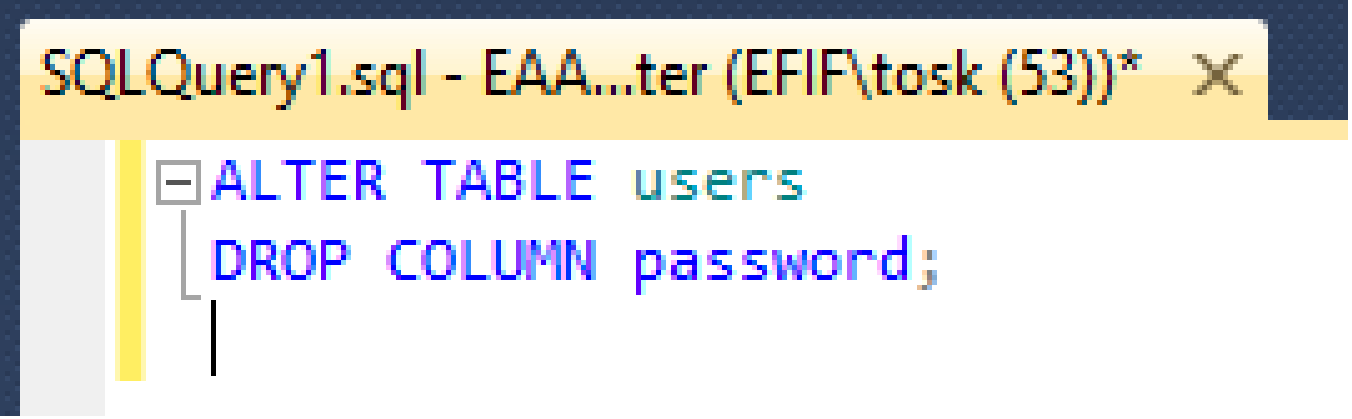


```
ALTER TABLE users
ADD password nvarchar(50);

INSERT INTO users(name, age, status, password)
VALUES ('Mary', 45, 'B', 'pass123');
```



```
> db.users.insert({
... name: "Mary",
... age: 45,
... status: "B",
... password: "pass123"});
WriteResult({ "nInserted" : 1 })
> _
```

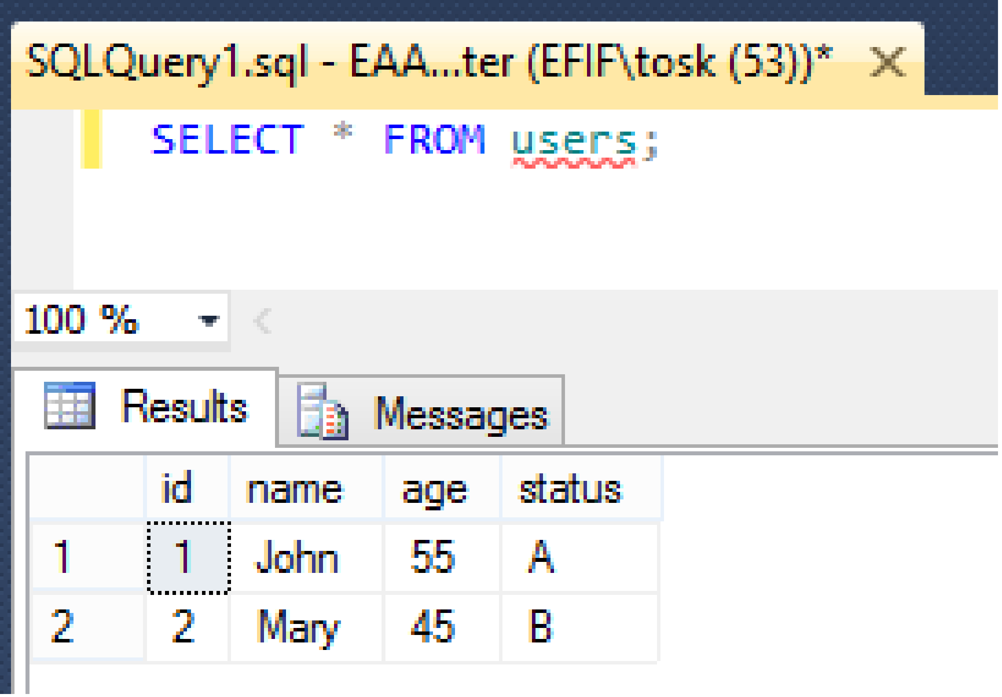
Alter



```sql
ALTER TABLE users
DROP COLUMN password;
```

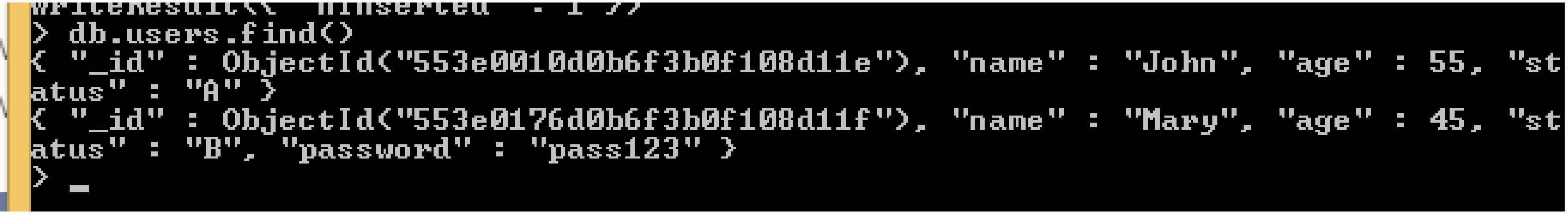
Collections do not describe or enforce the structure of its documents; i.e. there is no structural alteration at the collection level.

Drop column from table/collection



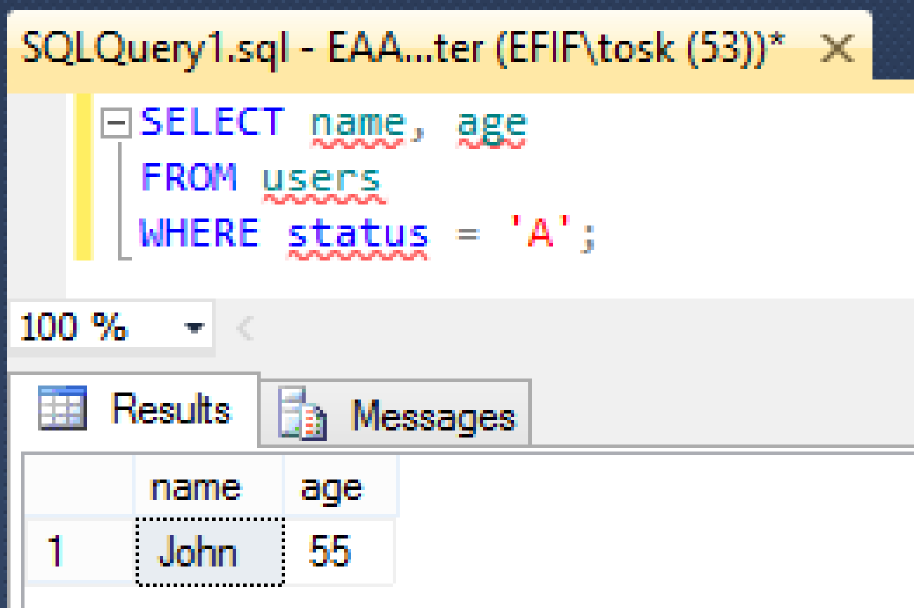
```sql
SELECT * FROM users;
```

	id	name	age	status
1	1	John	55	A
2	2	Mary	45	B



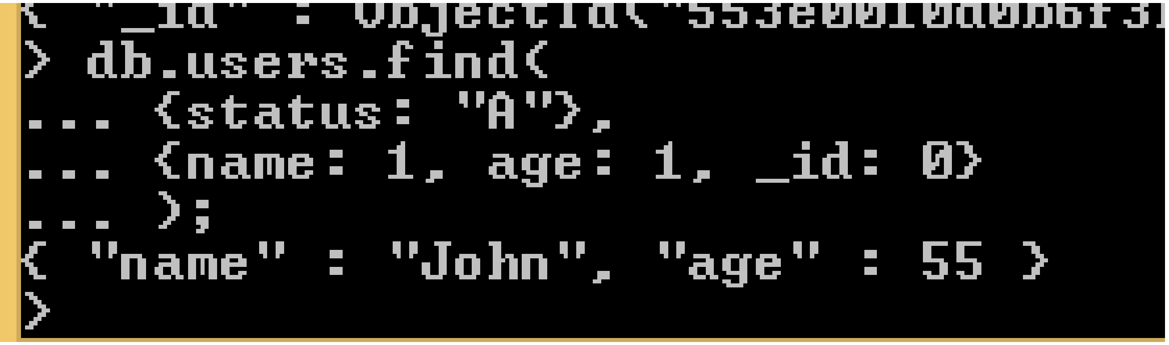
```
WriteResult({ "nInserted" : 1 })
> db.users.find()
{ "_id" : ObjectId("553e0010d0b6f3b0f108d11e"), "name" : "John", "age" : 55, "st
atus" : "A" }
{ "_id" : ObjectId("553e0176d0b6f3b0f108d11f"), "name" : "Mary", "age" : 45, "st
atus" : "B", "password" : "pass123" }
> _
```

SELECT



```sql
SELECT name, age
FROM users
WHERE status = 'A';
```

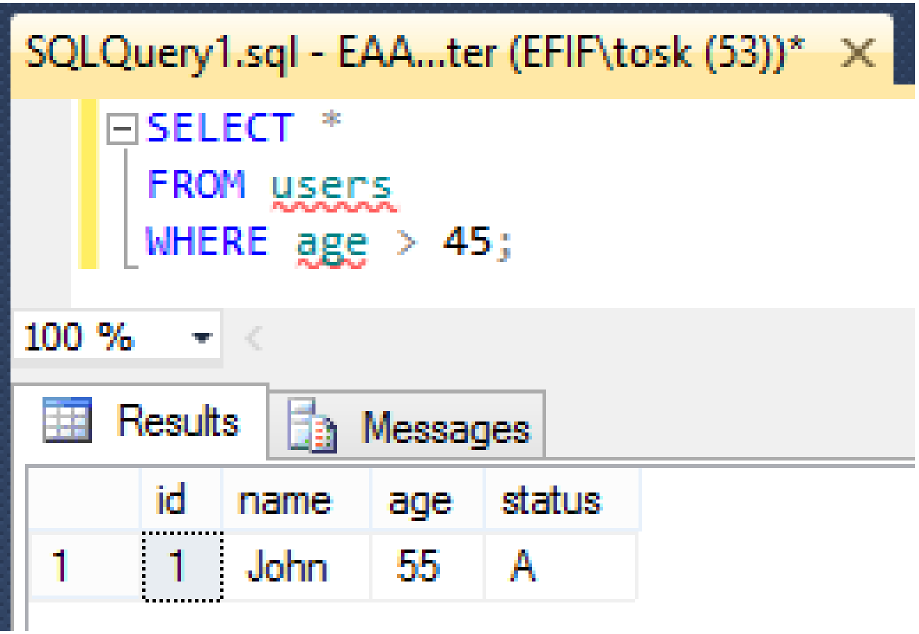
	name	age
1	John	55



```
{ "_id" : ObjectId("553e0010d00b6f31
> db.users.find(
... {status: "A"},
... {name: 1, age: 1, _id: 0}
... );
{ "name" : "John", "age" : 55 }
>
```

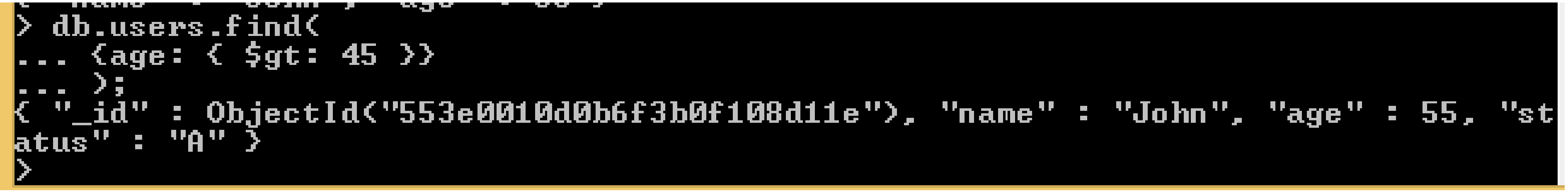
SELECT

BUSINESS ACADEMY
AARHUS

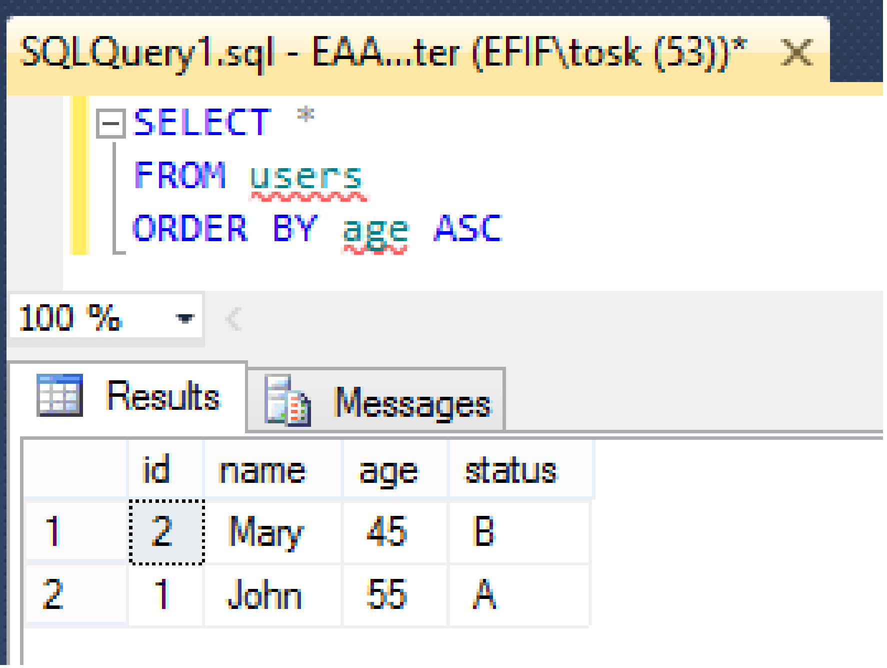


```sql
SELECT *
FROM users
WHERE age > 45;
```

	id	name	age	status
1	1	John	55	A

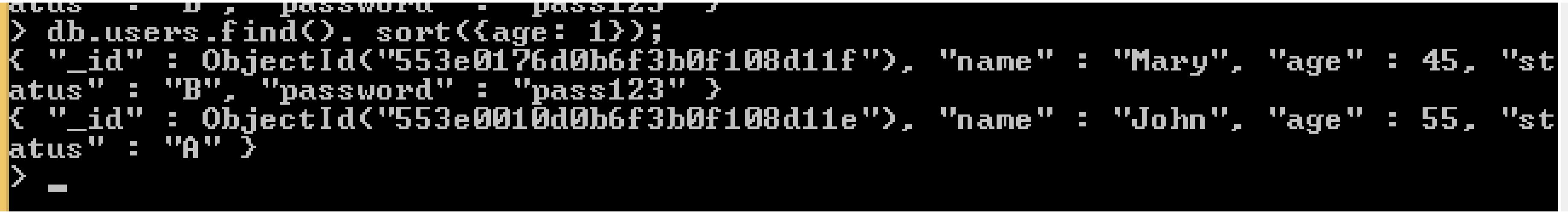


```
> db.users.find(
... {age: { $gt: 45 }}
... );
{ "_id" : ObjectId("553e0010d0b6f3b0f108d11e"), "name" : "John", "age" : 55, "st
atus" : "A" }
>
```



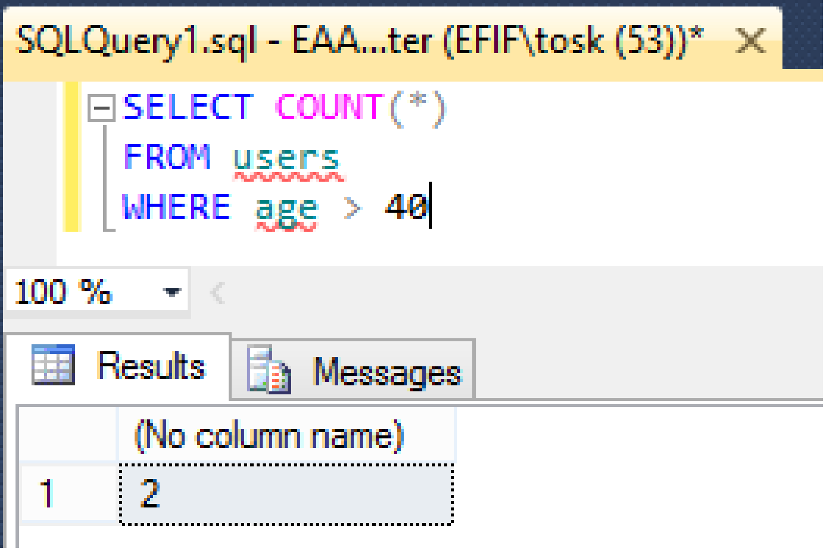
```
SELECT *
FROM users
ORDER BY age ASC
```

	id	name	age	status
1	2	Mary	45	B
2	1	John	55	A



```
atus" : "B", "password" : "pass123" }
> db.users.find(). sort({age: 1});
{ "_id" : ObjectId("553e0176d0b6f3b0f108d11f"), "name" : "Mary", "age" : 45, "st
atus" : "B", "password" : "pass123" }
{ "_id" : ObjectId("553e0010d0b6f3b0f108d11e"), "name" : "John", "age" : 55, "st
atus" : "A" }
> _
```

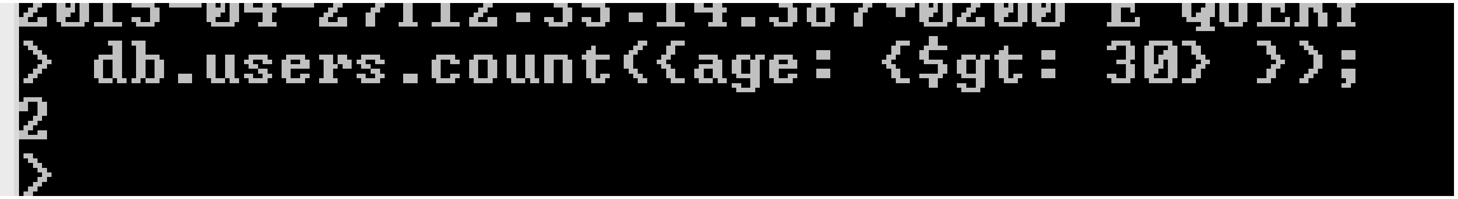
ORDER BY



SQLQuery1.sql - EAA...ter (EFIF\tosk (53))*

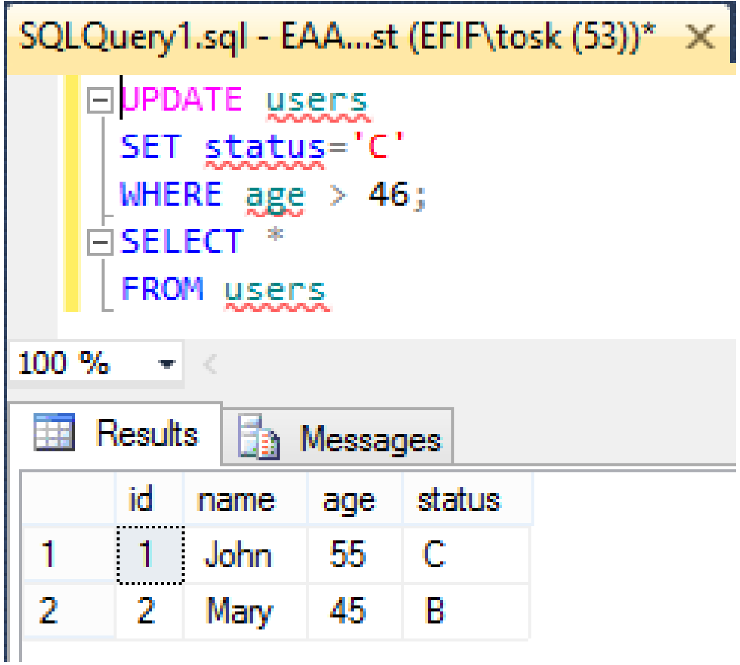
```sql
SELECT COUNT(*)
FROM users
WHERE age > 40
```

```
   (No column name)
1  2
```



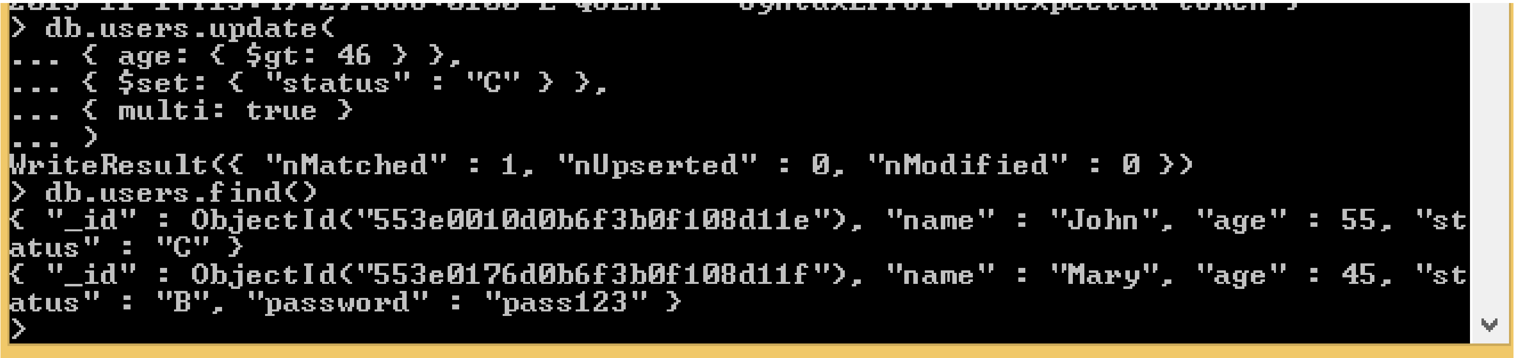
```
2015-04-27T12:35:14.367+0200 E QUERY
> db.users.count({age: {$gt: 30} });
2
>
```

Count



```sql
UPDATE users
SET status='C'
WHERE age > 46;
SELECT *
FROM users
```

	id	name	age	status
1	1	John	55	C
2	2	Mary	45	B



```
> db.users.update(
... { age: { $gt: 46 } },
... { $set: { "status" : "C" } },
... { multi: true }
... )
WriteResult({ "nMatched" : 1, "nUpserted" : 0, "nModified" : 0 })
> db.users.find()
{ "_id" : ObjectId("553e0010d0b6f3b0f108d11e"), "name" : "John", "age" : 55, "st
atus" : "C" }
{ "_id" : ObjectId("553e0176d0b6f3b0f108d11f"), "name" : "Mary", "age" : 45, "st
atus" : "B", "password" : "pass123" }
>
```

Update

Spend 20 minutes studying the topic that you get on a piece of paper and prepare a short presentation/explanation of what it means. The presentation should take up to 5 minutes. You will be presenting it to three of your class mates later today.

Use examples, drawings, images or whatever makes the explanation easier to understand. Powerpoints or other tools are also allowed.

Exercise part 1

Now you are going to find one person in class who has the same topic as you. Prepare your presentations, and learn from each other. Use the inspiration to add to you own presentation.

Exercise part 2

Make groups of four people with only persons who have a different subject than you. Now take turns presenting your topic. Each topic gets appr. 5 minutes, including (lots of) questions from the co-students.

Order:

Sharding
Indexing
Replication
Capped collections

Exercise part 3

The (Common) Database Technologies

Relational
Data**B**ase
Management
System

ORACLE
DATABASE

MySQL

PostgreSQL

Microsoft
SQL Server

Not
Only
SQL

mongoDB

redis

riak

APACHE
HBASE

Neo4j

CouchDB
relax

Key-Value Store redis riak memCached

Graph Neo4j INFINITEGRAPH®

Document Store mongoDB CouchDB relax RAVENDB

Column Family APACHE HBASE cassandra HYPERTABLE INC

NoSQL Database Types

mongoDB

Google, Facebook, ebay, Expedia, Adobe, MTV, EA, The New York Times, GAP, craigslist etc.

CouchDB
relax

BBC

RAVENDB

EB Games, Pluralsight, Microsoft Azure

Document Store

- Very similar to document store
- With a key-value store we can only access an aggregate by lookup based on its key
- With document databases, we mostly expect to submit some form of query based on the internal structure of the documents; this might be a key, but it's more likely to be something else

Key-Value store

Key-Value store

redis

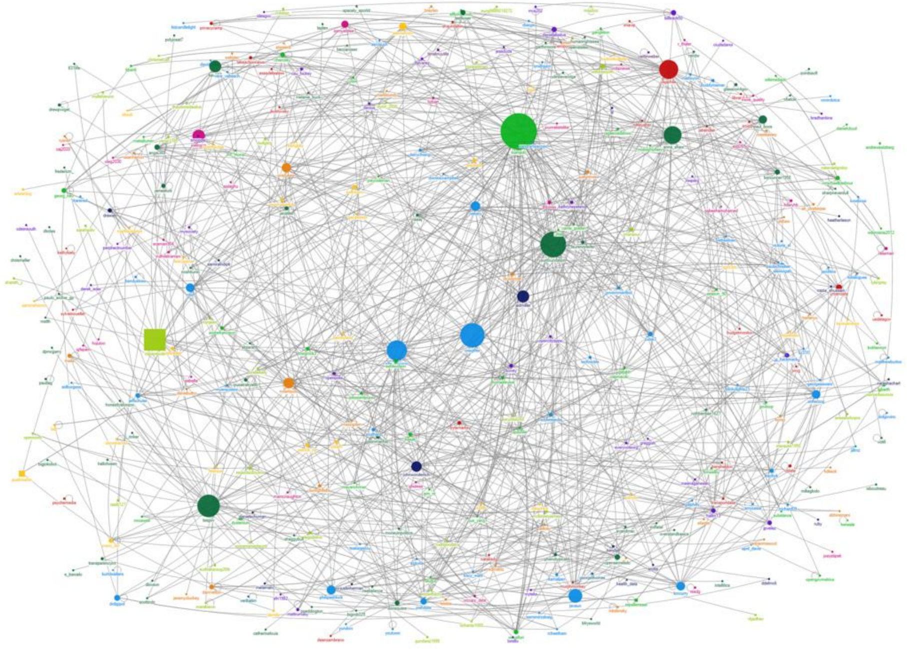
Twitter, GitHub, Pinterest, Snapchat, Craigslist, StackOverflow, Flickr, Airbnb, Tumblr

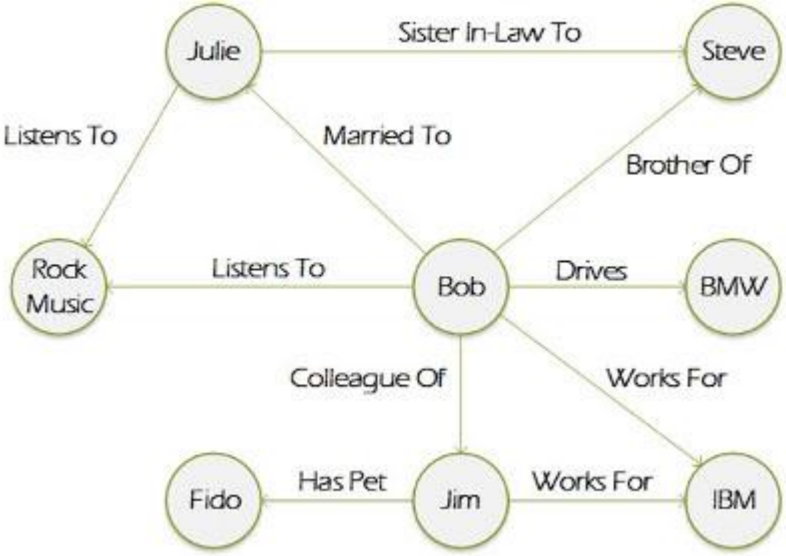
memCached

YouTube, Reddit, Facebook, Twitter, Wikipedia

riak

Bet365, McAfee, Virgin America, Yahoo Japan, The Weather Channel





Julie
Sister In-Law To
Steve
Listens To
Married To
Brother Of
Rock Music
Listens To
Bob
Drives
BMW
Colleague Of
Works For
Fido
Has Pet
Jim
Works For
IBM

Graph store

- Graph data structure of nodes connected by edges
- We can ask questions such as ”find the books in the Databases category that are written by someone whom a friend of mine likes”
- Graph databases specialize in capturing this sort of information – but on a much larger scale than a readable diagram could capture
- This is ideal for capturing any data consisting of complex relationships such as social networks and product preferences

Graph store

Graph databases

FlockDB	Twitter
Neo4j	ebay, LinkedIn, Lufthansa, TomTom, telenor, hp
INFINITEGRAPH®	

Column Family

- One of the early and influential NoSQL databases was Google’s BigTable
- Most databases have a row as a unit of storage which, in particular, helps write performance
- However, there are many scenarios where writes are rare, but you often need to read a few columns of many rows at once
- In this situation, it’s better to store groups of columns for all rows as the basic storage unit

Column Family

APACHE HBASE

Adobe, Facebook, Meetup, Twitter, Yahoo

cassandra

Adobe, AOL, BlackBerry, Call of Duty, Dell, Disney, ebay, GitHub, Hotels.com, IBM, Instagram, Microsoft Azure, Spotify, Twitter, Unity, Walmart

HYPERTABLE INC

Ebay, yelp

Create an account and get a working version of MongoLab up and running.

(Create new subscription -> Single-node -> Sandbox)

In MongoLab create a new collection and at least 5 new documents. At least one document has to contain an array, and one document has to contain another document/object.

Then test out both list view and table view.

If you have more time, play around and get to know MongoLab better.

Exercise

RoboMongo – a fast, shell centric MongoDB GUI that supports Windows, MacOS and Linux

MongoChef – free non-commercial for Windows, Mac and Linux

MongoVue – a desktop GUI for the Windows platform. Basic features are free

MongoHub – native Mac GUI

MongoTools

Check your school e-mail

Answer the questions about the school and classes

Survey

Create a database for a blog

Create the blogpost collection

Insert three blogposts with tags

Retrieve the blogposts via relevant search criterias (min. 5).

Exercise 4

Next time

Read:

- NoSQL Distilled:
  - Chapter 2 – Aggregate Data Models
  - Chapter 9 – Document Databases
  - Chapter 13 – Polyglot Persistence
  - Chapter 15 – Choosing Your Database

We will be looking more into Polyglot Persistence

We will using MongoDB even more

I will introduce the subjects for the exam

You will also get an exam question example that we practice during class.

Lectures till 14:00