

Document Databases: Practice

Seminar 3 of *NoSQL Databases* (PA195)

David Novak & Vlastislav Dohnal

Faculty of Informatics, Masaryk University, Brno

MongoDB Shell



stratus

- **Connect** to a running MongoDB database: [doc](#)
 - run **mongo shell**: `$ mongo local`
- MongoDB shell **basic** features & commands
 - use `tab` for command **completion**
 - use `up/down` arrows for command **history**
 - use `ctrl+R` for reverse **command search**
 - JavaScript syntax
 - `var record = { "_id": 1, "name": "david" }`

> help

> show dbs

> use local

To create a new database, just type:
use newdbname

Insert



Follow: <http://docs.mongodb.org/manual/core/write-operations-introduction/>

Or <http://docs.mongodb.org/manual/applications/crud/>

```
db.users.insert (  ← collection
  {
    name: "sue",    ← field: value
    age: 26,        ← field: value
    status: "A"     ← field: value
  }                } document
)
```

```
db.inventory.insert( { _id: 10, type: "misc",
item: "card", qty: 15 } )
```

- Inserts a document with three fields into collection **inventory**
 - User-specified **_id** field

Task 1



- **Create** 5-10 JSON objects according to the **schema**
 - of all types, i.e. "food", "snacks", "accessories", "misc"
 - Use an online editor + **validator**:
<http://www.jsonschemavalidator.net/>
- **Insert** these JSON documents into your **database**
 - into collection `inventory`

Task 2



- Insert these items into the database:

```
var macdonalds = [  
  {item: "burger", type: "snacks", price: 0.99, qty: 200 },  
  {item: "coke", type: "food", price: 1.99, qty: 500 },  
  {item: "french fries", type: "snacks", price: 1.0, qty: 5},  
  {item: "six pack", type: "food", price: 11.99, qty: 60 },  
  {item: "cup", type: "accessories", price: 0.05, qty: 1000,  
    producer: { address: '123 Street', company: 'ABC123' } },  
  {item: "straws", type: "accessories", price: 0.9, qty: 300,  
    producer: { company: 'ABC123', address: '123 Street' } }  
]
```

```
db.inventory.insert(macdonalds)
```

- Update items “burger” and increment quantity by 50.

Querying



- A MongoDB **query**:
 - Targets a specific **collection** of documents
 - Specifies **criteria** that identify the returned documents
 - May include a **projection** to **specify** returned **fields**
 - May impose limits, sort, orders, ...
- Basic query - all documents in the collection:

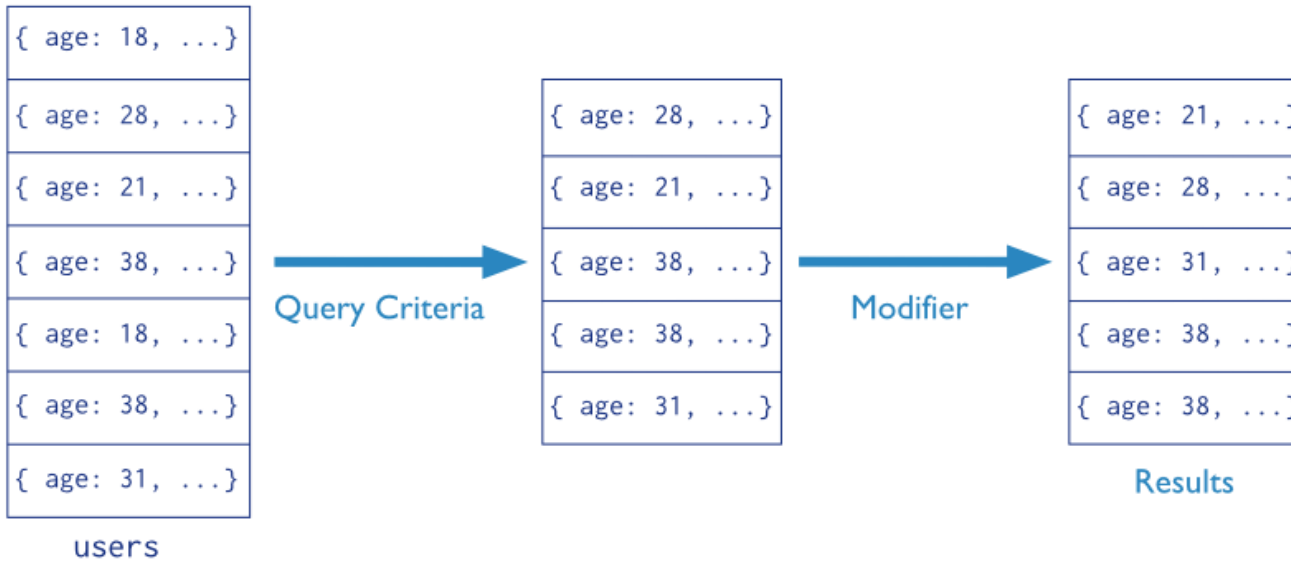
```
db.inventory.find()
```

```
db.inventory.find( {} )
```

Querying (2)



Collection Query Criteria Modifier
`db.users.find({ age: { $gt: 18 } }).sort({age: 1 })`



- Mongo query language

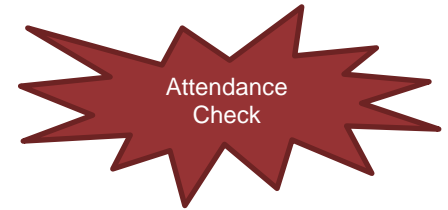
Querying (3)



```
db.inventory.find({ type: "snacks" })
```

- All documents from **inventory** where the **type** field has the value **snacks**

- Copy&paste the response to the file `find-snacks.txt` and upload to the IS's vault!



```
db.inventory.find({ type: { $in: [ 'food', 'snacks' ] } })
```

- All **inventory** docs where the **type** field is either **food** or **snacks**

```
db.inventory.find( { type: 'food', price: { $lt: 9.95 } } )
```

- All ... where the **type** field is **food** and the **price** is **less than 9.95**

Querying (6): Nested Values



```
db.inventory.find( { producer: {  
                    company: 'ABC123',  
                    address: '123 Street'  
                } } )
```

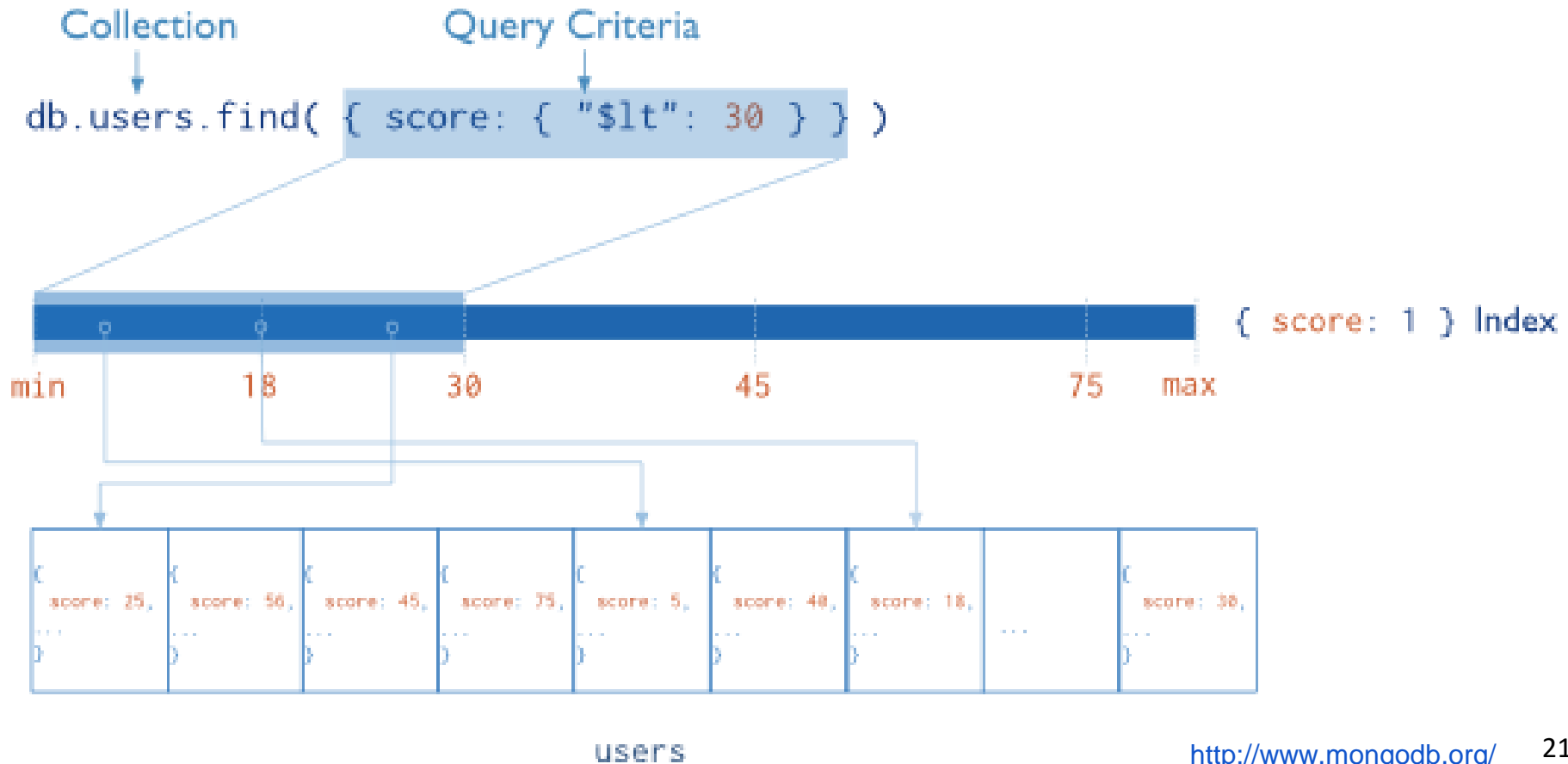
- All docs where the value of the field **producer** is a **subdocument** that contains only the field **company** with the value **ABC123** and the field **address** with the value **123 Street**, in the exact order

```
db.inventory.find( { 'producer.company':  
'ABC123' } )
```

- All docs where the field **producer** is a **subdocument** containing field **company** with value **ABC123** (and may contain other fields),₁₉



Indexes: Example of Use





Index Types by Attributes

- **Default: `_id`**
 - Exists by default
 - If applications do not specify `_id`, it is created.
 - Unique
- **Single Field**
 - **User-defined** indexes on a single field of a document
- **Compound**
 - User-defined indexes on **multiple** fields
- **Multikey index**
 - To index the content stored in **arrays**
 - Creates separate **index entry** for **each** array **element**



Index Types by Technique

- **Ordered Index**
 - B-Tree (see above)
- **Hash Indexes ("hashed")**
 - **Fast** $O(1)$ indexes the hash of the value of a field
 - Only **equality** matches
- **Geospatial Index**
 - "2d" = use **planar geometry** when returning results
 - For data representing points on a two-dimensional plane
 - "2dsphere" = **spherical** (Earth-like) geometry
 - For data representing longitude, latitude
- **Text Indexes ("text")**
 - Searching for **string** content in a collection

Indexes: Management



<https://docs.mongodb.org/manual/core/indexes-introduction/>

- **Basic command:**

```
db.inventory.createIndex( { type: 1 } )
```

- The number controls ordering (1:asc, -1:desc)

- **Check indexes & create hashed index**

```
db.inventory.stats()
```

```
db.inventory.dropIndex({type:1})
```

```
db.inventory.createIndex({type:"hashed"})
```

```
db.inventory.dropIndex({type:"hashed"})
```


Indexes: Management (2)



- Query **execution plan** (is **index** used?):

```
db.inventory.find({item: "burger"}).explain()
```

- In **real** application, **all queries** should use indexes

MongoDB Management



- **Installation & running**
 - `service mongod start/stop/status`
- **Configuration file**
 - `/etc/mongod.conf`
- **Data files**
 - `dbpath=/var/lib/mongo`
- **Log file**
 - `logpath=/var/log/mongodb/mongod.log`
- **MongoDB tools:**
 - `mongo<TAB>`
- **Admin GUI interfaces: [link](#)**

Task 3: Import Large Data



- generate JSON data:
 - <http://jsongen.pykaso.net/>
- or use generated file at
~/mongo/people.json
 - or download from [study materials](#)

Generate GPS
coordinates
(GeoJSON)

```
{
  "_id" : "%index%",
  "surname" : "%surname%",
  "fullname" : "%fullname%",
  "email" : "%email%",
  "salary": "%randFloat(10,50000)%",
  "address" : {
    "city" : "%name%",
    "street" : "%name%",
    "number" : "%randInt(0,50)%",
    "location" : {
      "type": "Point",
      "coordinates": [
        "%randFloat(0,180)%",
        "%randFloat(0,90)%"
      ]
    }
  }
}
```

Task 3 (cont.)



```
mongoimport --db <name> --collection <col> --  
type json --file <path> --jsonArray
```

- **database:** <name> (use "local")
- **collection:** <col> (use "people")
- **file type:** JSON (default)
- **file path:** <path> (use "~/mongo/people.json")
- assuming that the file contains an array of JSON documents



Task 4: Querying

- Find people living in the city “Andrew”
- Find people with salary greater than 3000
 - list their names and salary only
- Find people with email from the domain “kaplan.il”
- Count records in people collection (it is 10,000)
 - count distinct last names (it is 640)

Using an aggregate function?

Copy & paste (at least 3) queries and their responses to the file `find-people.txt` and upload to the IS's vault!

Attendance
Check

Geospatial Indexes



- **GeoSpatial** index and queries

```
db.people.ensureIndex({"address.location": "2dsphere"})
```

```
db.people.find(  
  { "address.location": { $near :  
    { $geometry :  
      { type : "Point" ,  
        coordinates : [ 128.4, 48.13 ]  
      }, $maxDistance: 10000  
    }  
  }  
})
```

In meters for
GeoJSON type

Connectors from Languages



- Python

- **Pymongo**: basic **operations** with the database
 - like MongoDB shell encapsulated in Python
- **Mongoengine**: Object-Document **Mapper**
 - transparent **storing** of object data
 - <http://mongoengine.org/>

- Example in

```
# ~/mongo/student.py  
# ./student.py --help
```

<https://is.muni.cz/auth/el/fi/podzim2020/PA195/um/seminar-3/student.py>

References

- I. Holubová, J. Kosek, K. Minařík, D. Novák. Big Data a NoSQL databáze. Praha: Grada Publishing, 2015. 288 p.
- Sadalage, P. J., & Fowler, M. (2012). NoSQL Distilled: A Brief Guide to the Emerging World of Polyglot Persistence. Addison-Wesley Professional, 192 p.
- RNDr. Irena Holubova, Ph.D. MMF UK course NDBI040: Big Data Management and NoSQL Databases
- MongoDB Manual: <http://docs.mongodb.org/manual/>