UV Distributed Information Management
Summer semester 2024

Assignment 02

Summary:

Deadline: May 28, 2024, 11:55 pm (aka 23:55) CET.
Extended Deadline: June 04, 2024, 11:55 pm (aka 23:55) CET.
Submission: Submit a compressed archive (e.g., a `zip` or a `tar.gz` file) that contains your Python3 code and your answers to the questionnaire via Blackboard.
Grading: 50% Python3 code, 50% answers (incl. meeting; cf. Section 5 for details).

---

1 Introduction

The purpose of this assignment is to get in touch with a widely used document-based- NoSQL database system (DBS), namely **MongoDB** [1], which is used in many modern applications. Similar to PostgreSQL, MongoDB [2] is source-available [3], rather easy to install and use, and also accessible using the Python3 programming language [4] (i.e., using the `pymongo` [5] module). In contrast to PostgreSQL, MongoDB is based on the document-based data model [6].

1.1 Grading & Submission

For your convenience, Section 5 contains the grading scheme that will be applied for this assignment.

Please submit your final Python3 code **and** your answers to the questionnaire until May 28, 2024, 11:55 pm (aka 23:55) CET via Blackboard [7] (late submission until June 04, 2024, 11:55 pm (aka 23:55) CET). Furthermore, please keep in mind that the exams contribute 40% to your final grade, hence you need to submit at least one assignment (partially) to pass the course.

1.2 Formatting Conventions

Commands for the Linux command-line tool (`terminal`) [8] and the command-line tool of MongoDB (`mongosh`) are written in `TrueType` font [9]. In addition, all commands are in a box that specifies the used command-line tool at the beginning of the title (separated by a dash –, i.e., **`terminal`** for Linux and **`mongosh`** for MongoDB). Listing 1 shows an example command executed in the Linux `terminal`.

The Linux `terminal` shows a prefix `dbtutorial@database-tutorial:~#` that consists of

[1] The name refers to the **humongous** amounts of data MongoDB is able to manage.
[2] MongoDB: `https://de.wikipedia.org/wiki/MongoDB`
[3] Source-available software: `https://en.wikipedia.org/wiki/Source-available_software`
[4] The Python programming language: `https://en.wikipedia.org/wiki/Python_(programming_language)`
[5] `https://pymongo.readthedocs.io/en/stable/` and `https://pypi.org/project/pymongo/`
[6] Document-oriented database: `https://en.wikipedia.org/wiki/Document-oriented_database`
[7] Blackboard: `https://elearn.sbg.ac.at`
[8] Command-line tool: `https://en.wikipedia.org/wiki/Command-line_interface`
[9] `TrueType` font: `https://en.wikipedia.org/wiki/TrueType`

- the name of the user that executes the command; `dbtutorial` is the default user of the virtual machine (VM) (this may be different if you do not use the given VM),
- the name of the machine; `database-tutorial` is the name of the given VM, and
- a delimiter that separates the actual command from the "`user@machine`" string; "`:~#`" is the default delimiter of the given VM (where ~ denotes the current directory).

Moreover, we use a color code for readability:

- A **command** itself (which is to be copied) is depicted in bold, solid **`black`**.
- **Prefixes**, which are **not** part of a command itself, are shown in `gray`.
- **Outputs**, which are the result of a command that has been executed, are shown in `green`.

Listing 1: **`terminal`** – Show directories.

```
dbtutorial@database-tutorial:~# ls -l
total 32
drwxr-xr-x 2 dbtutorial dbtutorial 4096 Mar 31 15:43 Desktop
drwxr-xr-x 2 dbtutorial dbtutorial 4096 Feb 28  2022 Documents
drwxr-xr-x 2 dbtutorial dbtutorial 4096 Feb 28  2022 Downloads
drwxr-xr-x 2 dbtutorial dbtutorial 4096 Feb 28  2022 Music
drwxr-xr-x 2 dbtutorial dbtutorial 4096 Feb 28  2022 Pictures
drwxr-xr-x 2 dbtutorial dbtutorial 4096 Feb 28  2022 Public
drwxr-xr-x 2 dbtutorial dbtutorial 4096 Feb 28  2022 Templates
drwxr-xr-x 2 dbtutorial dbtutorial 4096 Feb 28  2022 Videos
```

Contrarily, Listing 2 exemplifies a command in MongoDB's command-line tool, i.e., `mongosh`, which stands for **Mongo shell**. It shows an example that executes the command `show dbs` on a database named `test`:

Listing 2: **`mongosh`** – Show all databases.

```
test> show dbs
```

For commands that are to be executed in the `mongosh` terminal, the non-bold prefix `test>` denotes the database we are currently connected to (cf. Section 2.3 for more details). The MongoDB query language (MQL) is different from SQL and keywords like `show` and `db` are typically **not** capitalized. Furthermore, MongoDB has a dedicated command-line tool called `mongoimport` to import data into the database. Additional information can be found in the supplementary material given in Section 4.

Python3 [10] code is simply wrapped in a box without specifying any command-line tool, and we provide a `.py` file that contains the Python3 code (in order to make using the template easier).

1.3 Support

Remark: Please notify the instructor as soon as possible if this assignment description is (partially) unclear or if there is a problem with the submission.

If you have trouble understanding this assignment, please use one of the following communication channels to get help (in this order):

1. **Lecture:** wednesdays 01:00 - 02:30 pm CET (exception: lecture-free periods).
2. **Slack:** `https://dbteaching.slack.com/archives/C06MFP830GH` (I will check regularly and do my best to reply fast, but please do not expect me to be available 24/7).
3. **Email:** dkocher@cs.sbg.ac.at (as a last resort).

We recommend to start the assignment early. In case of a problem, it is easier for the instructor and other students to provide help in time if you identify problems early. e

[10] The Python Programming language: `https://en.wikipedia.org/wiki/Python_(programming_language)`

2 Assignment Description

We **highly recommend** that you read this section (including all subsections) to the end before you start working (this should be less error-prone).

Similar to Assignment 1, we split this assignment into five parts and only parts 1 (depending on your choice), 4, and 5 contribute to the overall grade of this assignment. We recommend to follow these steps and refer to the corresponding sections for more details.

1. Set up and configure MongoDB; cf. Section 2.2 and Section 2.3.
2. Create collections and fill them with data, cf. Section 2.4.
3. Familiarize yourself with MongoDB, cf. Section 2.5.
4. Write an example application that accesses the database in Python3; cf. Section 2.6.
5. Answer the questionnaire; cf. Section 2.7.

2.1 Introduction to MongoDB and JSON

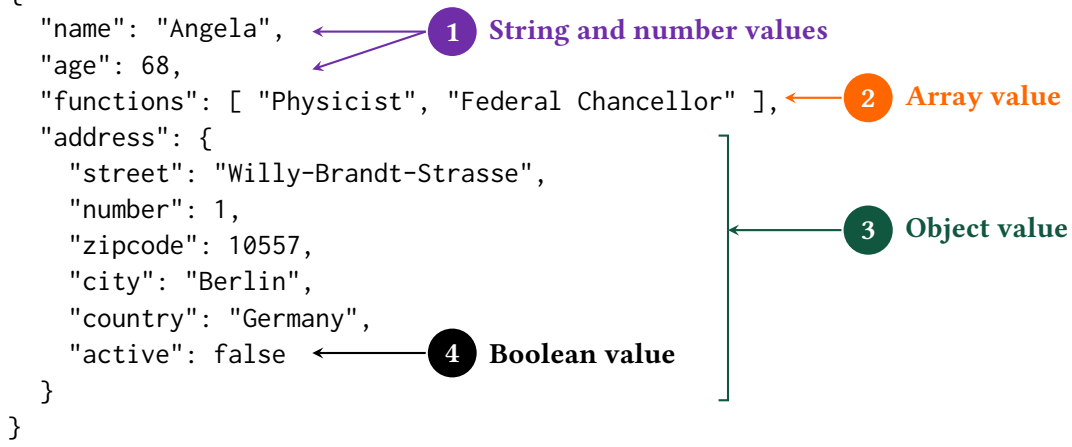
MongoDB is a NoSQL [11] database system that complies to the document-based data model. Since we did not cover many details in the lecture with regard to this data model, we will briefly introduce MongoDB and the underlying data model in more detail.

From the lecture, we know that the document-based data model stores data in a semi-structured [12] and nested format, so-called **documents**. As a document format, MongoDB uses the JavaScript Object Notation (JSON) [13], which is a human-readable text format for data structures that consists of (possibly nested) key-value pairs. A JSON document contains a JSON object (enclosed by curly braces: `{ . . . }`). Keys and values in JSON are separated by a colon (`:`), and multiple key-value pairs are separated by a comma (`,`). The curly braces are also used to structure the JSON object. In fact, any data enclosed by curly braces is a JSON object **itself**. Keys are strings [14] (i.e., enclosed by double quotes: "…") but values can be numbers (cf. 1), strings (cf. 1), booleans [15] (`true` or `false`; cf. 4), arrays [16] (an ordered, comma-separated list of values enclosed by squared brackets, `[ 0, 1, 2, . . . , 9 ]`; cf. 2), or JSON objects [17] themselves (again enclosed by curly braces; cf. 3). The fact that a value can be a JSON object itself allows to nest JSON objects arbitrarily (i.e., introduce a notion of hierarchy in a JSON document, cf. 3).

In contrast to the relational model of Assignment 1, MongoDB does not maintain tables but so-called **collections**. A collection stores multiple **documents** in JSON format and can be regarded as the equivalent of a table in relational database systems. Although MongoDB uses a binary JSON format called BSON [18] as internal representation, it is sufficient for this assignment to understand the concept of the JSON format. For interested students, a link to a detailed description of MongoDB's document format and the grouping as collection is provided in Section 4.

[11] NoSQL: `https://en.wikipedia.org/wiki/NoSQL`
[12] Semi-structured data: `https://en.wikipedia.org/wiki/Semi-structured_model`
[13] The JavaScript Object Notation: `https://en.wikipedia.org/wiki/JSON` and `https://www.json.org/json-en.html`
[14] The string data type: `https://en.wikipedia.org/wiki/String_(computer_science)`
[15] The boolean data type: `https://en.wikipedia.org/wiki/Boolean_datatype`
[16] The array data type: `https://en.wikipedia.org/wiki/Array_data_structure`
[17] The object data type: `https://en.wikipedia.org/wiki/Object_(computer_science)`
[18] The Binary JSON format: `https://en.wikipedia.org/wiki/BSON`

JSON Example:



2.2 MongoDB Setup

Similar to Assignment 1, there are two possible ways to set up and configure MongoDB in order to solve this assignment:

MongoDB Installation Install MongoDB on your own system (or virtual machine) using an operating system of your choice. This implies that you "pollute" your system with this installation and that the instructor may need additional information on your particular setup to provide help (since we mostly work on Linux systems). Nonetheless, we want to emphasize that it is an **excellent exercise** for students to perform the installation of such a system on their own (at least once in their career). If you choose this option, you can jump to Section 2.2.1.

MongoDB using a VM Configure a pre-installed MongoDB in a virtual machine (VM) that runs Debian Linux. This implies that you will not experience the process of installing MongoDB on your own, but it may be easier for the instructor to help since the VM runs Debian Linux (and, ideally, the setup is reproducible). On the one hand, it may be cumbersome if you do not have any experience with Linux systems (and VMs), but on the other hand it may also be a nice opportunity to familiarize yourself and play around with a Linux system (and a VM). To continue with this option, you can jump to Section 2.2.2.

From our experience, the installation of MongoDB should not cause many troubles. Hence, we encourage everyone to give it a try. You can choose your preferred way and the choice itself will not influence your grade in any way. However, make sure that you can answer questions regarding your choice.

2.2.1 MongoDB Installation

Remarks: (i) If you decide to use the VM image, you can directly jump to Section 2.2.2. MongoDB is already pre-installed. (ii) Please note that MongoDB's command-line tools may have to be downloaded and installed separately, for example, on Windows systems. (iii) Please use the command-line tool of your operating system to call MongoDB's command-line tools. On Windows systems, for example, use the `cmd.exe` or the `Powershell` (and navigate into the directory that contains MongoDB's command-line tools, i.e., the directory that you chose during installation).

The first step is to install a MongoDB server (Community Edition) locally on your (possibly virtual) machine. This should be possible on almost all operating systems, but the specific

steps may diverge. However, the documentation of MongoDB provides all the details on how to install MongoDB on different operating systems (cf. links given in Section 4). Once MongoDB is installed, you can create a database and import data in the JSON format. Like PostgreSQL, MongoDB also provides a graphical user interface named **Compass** [19]. Nonetheless, we recommend to use MongoDB's command-line tools, which allow you to interact with the database system using the command line. In particular, the `mongosh` command-line tool (aka `mongo` **shell**) provides the most basic way to use the database. Furthermore, you will need to use the `mongoimport` command-line tool that provides an easy way to import data given as JSON files into the database.

2.2.2 Virtual Machine Setup

Remark: We are going to reuse the virtual machine of Assignment 1. Therefore, you can directly jump to Section 2.3 if you solved Assignment 1 with the corresponding virtual machine.

We provide a so-called virtual machine (VM) **image**, which consists of a pre-installed Debian Linux as operating system and a pre-installed MongoDB instance. In this case, you may have to familiarize yourself with the concept of a VM and how to host the corresponding image (see Assignment 0 for details). Essentially, a virtual machine [20] is a software that is designed to provide you with a substitute of a real machine, that is, it runs another operating system on top of your regular operating system – it **virtualizes** the hardware of another system. For this assignment, it suffices to install a software named VirtualBox [21] that acts as a host for virtual machines. After you successfully installed VirtualBox, the VM image can be used to set up the virtual machine that can be used for this assignment. The image (as `ova` file) can be downloaded from a Google drive [22] and runs Debian Linux [23] as operating system with an installation of MongoDB.

After downloading the `debian-vm.ova` file, this file must be imported into VirtualBox. There are many online tutorials [24] on how to accomplish this and it primarily consists of two steps (see also Assignment 0): (1) Choose the image to import (navigate to the downloaded file) and (2) check/modify the settings of the virtual machine (typically no modifications are required, but sometimes, for example, you may have to disable the USB port). Once you click on the **Import** button, VirtualBox imports the image and sets up the virtual machine that can be used for this assignment. VirtualBox then shows a new VM on the left-hand side, which can be started/booted with a double-click.

For this VM, there exist two users: (1) `dbtutorial` and (2) `root`. If you are not familiar with the concept of a `root` user (or superuser), please check out the corresponding article [25] on Wikipedia. In essence, the `root` user has **all** privileges whereas the `dbtutorial` user has not. By default, we will work with the `dbtutorial` user, but we will temporarily switch to the `root` user if we need additional privileges. The password for both users is the same: `dbpwd1`

Once you logged into the VM and before you start working on the actual assignment, try to familiarize yourself a bit with Debian Linux (unless you already have experience using Linux systems and/or solved Assignment 0). For example, open a Linux `terminal` (aka Linux **shell** or **command-line tool**) and try some basic commands (some links can be found in the supplementary material, cf. Section 4). In particular, we will use the `mongosh` command-line tool (aka Mongo **terminal** or **shell**), which provides the most basic way to use MongoDB.

[19] MongoDB Compass: `https://www.mongodb.com/products/compass`
[20] Virtual machine: `https://en.wikipedia.org/wiki/Virtual_machine`
[21] VirtualBox: `https://www.virtualbox.org/manual/ch02.html`
[22] VM image for this assignment: `https://drive.google.com/file/d/1s6bp8TyzFfHL08EZaI1P4Qm6nnTG7mFA/view?usp=sharing`
[23] Debian Linux: `https://en.wikipedia.org/wiki/Debian`
[24] Virtual machine import: `https://docs.oracle.com/cd/E26217_01/E26796/html/qs-import-vm.html`
[25] The `root` user: `https://en.wikipedia.org/wiki/Superuser`

Remark: Note that you may not be able to use the typical `CTRL+C` and `CTRL+V` sequence to copy and paste between your regular system and the VM as the clipboard is not shared by default [26].

2.3 MongoDB Configuration

After the installation, the first step is to check whether the MongoDB daemon (or service) is running on your system. A **daemon** is a process that runs in the background. In the case of MongoDB, it manages the data and the requests [27]. This is most likely **not** the case and it is part of this assignment to (re-)start the MongoDB daemon named `mongod` as shown in Listing 3.

Remark: The `dbtutorial@database-tutorial:~$` (as well as `root@database-tutorial:~#`) at the beginning should already be displayed in your terminal and is not part of the command itself (which is `su -`). Do not get confused by the fact that nothing is shown when you type in the password (not even asterisks `*****`, which are often used for passwords). This is the default behavior of Linux and you will be asked again if the password is incorrect.

In Listing 3, we first confirm that the MongoDB daemon is **not** running (line 1) using the `systemctl` command. This is only a passive command, hence we can execute this command with the `dbtutorial` user. We observe that the state of our MongoDB daemon is `inactive (dead)`, i.e., it exists but is not running. Next, we need to execute an active command that **modifies** the state of our operating system by starting the MongoDB daemon. To this end, we must first switch to the `root` user (line 6) as the `dbtutorial` user does not have enough privileges to perform this action. Then, we use the `systemctl` command to (re-)start the MongoDB daemon (line 8). Line 9 then switches back to the `dbtutorial` user (as we do not want to stay in `root` mode).

Afterwards, we confirm that the MongoDB daemon is running (cf. Listing 4).

On Windows systems, one way to see whether the MongoDB daemon is running is to look for a process called `mongod` in the Task Manager [28].

Listing 3: **`terminal`** – (Re-)Start the MongoDB daemon.

```
1  dbtutorial@database-tutorial:~$ systemctl status mongod
2  ● mongod.service - MongoDB Database Server
3       Loaded: loaded (/lib/systemd/system/mongod.service; disabled; vendor . . . )
4       Active: inactive (dead)
5         Docs: https://docs.mongodb.org/manual
6  dbtutorial@database-tutorial:~$ su -
7  Password:
8  root@database-tutorial:~# systemctl restart mongod
9  root@database-tutorial:~# exit
```

Listing 4: **`terminal`** – Confirm that the MongoDB daemon is running.

```
1  dbtutorial@database-tutorial:~$ systemctl status mongod
2  ● mongod.service - MongoDB Database Server
3       Loaded: loaded (/lib/systemd/system/mongod.service; disabled; vendor . . . )
4       Active: active (running) since Tue 2023-05-02 13:12:12 CEST; 5s ago
5         Docs: https://docs.mongodb.org/manual
6     Main PID: 57756 (mongod)
7       Memory: 192.5M
8           CPU: 1.015s
9       CGroup: /system.slice/mongod.service
10                ÀÄ57756 /usr/bin/mongod –config /etc/mongod.conf
```

In this case, the `systemctl` command shows a lot of status information about the MongoDB

[26] Shared clipboards in VirtualBox: `https://www.youtube.com/watch?v=fqrJ7qlhJu0`
[27] The MongoDB daemon: `https://docs.mongodb.com/manual/reference/program/mongod/`
[28] Windows' Task Manager: `https://en.wikipedia.org/wiki/Task_Manager_(Windows)`

daemon, e.g., the starting time, a process ID (PID), used memory, and the likes. This information may be different on your system, and the most important information is that the MongoDB daemon is `active (running)`. Once the MongoDB daemon is running, we can connect to our local database without credentials using the MongoDB shell as shown in Listing 5.

Listing 5: **terminal** – Connect to the local MongoDB server without credentials.

```
dbtutorial@database-tutorial:~$ mongosh "mongodb://localhost"
```

If you configured your MongoDB server to use credentials and/or a custom port, the connection string [29] must be adjusted accordingly (cf. Listing 6).

Listing 6: **terminal** – Connect to the local MongoDB server with credentials (user: `dbtutorial`, password: `pass`) and a custom port (`4242`).

```
dbtutorial@database-tutorial:~$ mongosh "mongodb://dbtutorial:pass@localhost:4242"
```

After the connection has been established, we can start to interact with the database. Similar to PostgreSQL, MongoDB has a default database named `test`, and we can use the command shown in Listing 7 to display the database that is currently in use (i.e., the database we operate on; also indicated by the `test>` prefix).

Remark: The `test>` at the beginning should already be displayed in your `mongosh` terminal and is not part of the command.

Listing 7: **mongosh** – Show the database we are currently on.

```
test> db
test
```

This should report that we are currently on the database named `test`. To see all databases that exist on our MongoDB server, we can use the command shown in Listing 8 (the size may be slightly different on your system).

Listing 8: **mongosh** – Show all databases on our server.

```
test> show dbs
admin 8.00 KiB
config 12.00 KiB
local 8.00 KiB
```

In contrast to PostgreSQL, in MongoDB we can switch to a new database on the fly, meaning that MongoDB creates the database as soon as we fill the database with data. Therefore, we can use the commands shown in Listing 9 to switch to a new database named `assignment2` (and confirm it).

Listing 9: **mongosh** – Switch database (and verify it).

```
test> use assignment2
switched to db assignment2
assignment2> db
assignment2
```

Please use `assignment2` as database for this assignment (indicated by `assignment2>`).

[29] MongoDB's connection string: `https://docs.mongodb.com/manual/reference/connection-string/`

Remark: The `assignment2>` at the beginning should already be displayed in your `mongosh` terminal and is not part of the command.

2.4 Data Initialization

Now, we can proceed to fill our database with some data – by now it is empty. We can verify that it contains no data by executing the command shown in Listing 10, which shows all collections in our database (i.e., none). Therefore, we need to populate the database with some data.

Listing 10: **`mongosh`** – Show all collections in our database.

```
assignment2> show collections
```

In the course of this assignment, we will use data of two publicly available archives for scientific publications in the computer science domain:

- The DBLP Computer Science Bibliography [30], which provides open bibliographic information on computer science publications. This dataset consists of 1,984,049 JSON documents.
- A small portion of the arXiv repository [31], which is a document server for pre-prints of publications. This dataset contains 3 JSON documents.

First, we must download the data from our Nextcloud [32] and unzip it. Unlike in Assignment 1, we do not have to create tables (or documents) before we import the data. Instead, the structure of each single document is encoded in the document itself and we simply import the documents directly into the database. This is the first benefit of MongoDB's schema independence. We fill the database with the data of two plain files (each of which contains one JSON document per line), which are imported such that their documents fill the respective collections. Each single JSON file results in one collection that in turn consists of all JSON documents that are to be found in the corresponding JSON file.

MongoDB provides a command-line tool called `mongoimport` that can be used to import the data into the `assignment2` database as depicted in Listing 11 (note that `mongoimport` is called from the Linux terminal).

Remark: The easiest way to import the data is to open another terminal to execute the `mongoimport` commands. Then, you do not have to disconnect from your MongoDB server and can immediately continue after you successfully imported the data.

Listing 11: `terminal` – Import the plain JSON files (`user@machine` string shortened).

```
dbtut..@:~$ mongoimport --db assignment2 --collection dblp --file dblp.json
. . .
. . . 1984049 document(s) imported successfully. 0 document(s) failed to import
dbtut..@:~$ mongoimport --db assignment2 --collection arxiv --file arxiv.json
. . .
. . . 3 document(s) imported successfully. 0 document(s) failed to import
```

After the import, MongoDB provides feedback on the number of documents that have been imported, i.e., 1,984,049 and 3 documents habe been imported into the collections `dblp` and `arxiv`, respectively.

Remarks: (i) The `-file` option of these commands also works with full paths, for example, `-file "C:\Users\dkocher\Desktop\dblp.json"`. (ii) Importing the DBLP data into the corre-

[30] The DBLP Computer Science Bibliography: `https://dblp.uni-trier.de/`
[31] The arXiv repository: `https://arxiv.org/`
[32] Data for this assignment: `https://kitten.cosy.sbg.ac.at/index.php/s/sq6SNrpAyjQoNtt/download`

sponding collection will take some time, hence please wait for the commands to finish (MongoDB shows the current progress and whether the import was successful or not). (iii) On Windows systems, `mongoimport.exe` should be executed from the Windows command-line tool (`cmd.exe` or Powershell) and **not** via double click on the `mongoimport.exe` (it will close immediately since the parameters are missing). Therefore, please execute the Windows command-line tool in the directory that contains the `mongoimport.exe` file (or navigate into it using `cd`).

This creates two collections named `dblp` and `arxiv`, and we can confirm this by executing the `show collections` command in the `mongosh` command-line tool as shown in Listing 12. As expected, we observe that there exist two collections named `arxiv` and `dblp`.

Listing 12: **mongosh** – Show all collections in our database (after import).

```
assignment2> show collections
arxiv
dblp
```

2.5 Introduction to MQL

Once the data has been successfully imported into our `assignment2` database, try to further familiarize yourself with the `mongosh` command-line tool. Unlike PostgreSQL, MongoDB does not support SQL statements but uses the MongoDB query language (MQL). MQL is an intuitive query language based on the JSON format that is designed for application developers. MongoDB supports four so-called **CRUD operations: C**reate, **R**ead, **U**pdate, and **D**elete. In the course of this assignment, we will use read, create, and update operations. The MongoDB documentation often contains equivalent SQL statements for given MQL statements (cf. Section 4). Before we provide MQL queries that can be executed out of the box (cf. Listings 14– 17), we briefly cover some basics of the MongoDB query language.

First, the collection on which the operation should be executed must be specified. To refer to a specific collection, MongoDB uses a **Dot** notation [33]. For example, `db.dblp` refers to the DBLP collection in our database (DB). Similarly, we can execute an operation on this collection using the Dot notation by specifying the name of the operation. The query shown in Listing 13 executes the `find()` operation on the DBLP collection and uses the `pretty()` [34] operation to display the results in a human-readable format (otherwise each JSON document is printed as a single line, which is rather unreadable). Note that MongoDB does not return all documents immediately, but a cursor that can be used to iterate over the documents in the result set of the query (indicated by `Type "it" for more`). Consequently, we can retrieve additional results by typing `it` into `mongosh`.

Listing 13: **mongosh** – Execute the `find()` operation on the DBLP collection.

```
assignment2> db.dblp.find().pretty()
[
  {
    _id: ObjectId("595c2c37a7986c0872f266e7"),
    mdate: '2014-07-15',
    author: [ 'John Meyer' ],
    . . .
  }
]
Type "it" for more
```

The MQL query in Listing 13 corresponds to the SQL query `SELECT * FROM dblp`, i.e., we

[33] The Dot notation: `https://en.wikipedia.org/wiki/Property_(programming)#Dot_notation`
[34] The `pretty` operation: `https://www.mongodb.com/docs/manual/reference/method/cursor.pretty/`

request all documents of the DBLP collection (without any constraints). Similar to the `WHERE` clause in SQL, we can define one or more criteria to filter the documents before they are returned. This can be accomplished by passing a JSON object to the `find()` operation. In this JSON object, you are able to specify the criteria that must be met by a document in the result set of the query.

In the following, we provide **four** queries (cf. Listings 14– 17) that can be executed out of the box by entering them into the `mongosh` command-line tool (one after another; **without** trailing semi-colon):

Listing 14: **mongosh** – Query **Q1**.

```
assignment2> db.dblp.find({ "author": "Michael Stonebraker" }).pretty()
```

Listing 15: **mongosh** – Query **Q2**.

```
assignment2> db.dblp.find({
  "author": "Michael Stonebraker",
  "booktitle": "ICDE"
}).pretty()
```

Listing 16: **mongosh** – Query **Q3**.

```
assignment2> db.arxiv.aggregate({
  "$lookup": {
    "from": "dblp",
    "localField": "title",
    "foreignField": "title",
    "as": "arxivdblp"
  }
}).pretty()
```

Listing 17: **mongosh** – Query **Q4**.

```
assignment2> db.dblp.find({
  "author": "Michael Stonebraker",
  "booktitle": "ICDE"
}).explain()
```

In query **Q1**, we pass a JSON object (mind the enclosing curly braces) with a single key-value pair, `author: "Michael Stonebraker"`, to the `find()` [35] operation. Consequently, only documents that have been authored by "Michael Stonebraker" [36] are returned by MongoDB.

Query **Q2** passes a JSON object with two key-value pairs to the `find()` operation. This is similar to the `WHERE` clause of query **Q2** of Assignment 1: Only documents that satisfy **both** criteria (i.e., the documents that satisfy both key-value pairs) are returned.

MongoDB also supports a `JOIN`-like operation by linking documents of two (or more) collections and augmenting the joined documents of the collection that is specified first, i.e., `db.arxiv` (which is depicted in query **Q3**). Although it is not as intuitive as the `JOIN` operation in SQL, it follows the same principle: We specify the first collection using the Dot notation (`db.arxiv`) and then use the `aggregate()` operation to define the second collection (using `from: "dblp"`). In order to perform the join, we use `$lookup` [37], which adds a **new** field to each input document (note that we can also omit the double quotes around `$lookup` because this is a special key). The name of the new field (i.e., its key) is specified using the "`as`" field in our JSON object. The value of the new field contains all documents that satisfy the join criterion (i.e., have the same

[35] The `find` operation: `https://www.mongodb.com/docs/manual/reference/method/db.collection.find/`
[36] Michael Stonebraker: `https://en.wikipedia.org/wiki/Michael_Stonebraker`
[37] The `$lookup` operation: `https://docs.mongodb.com/manual/reference/operator/aggregation/lookup/`

title). To this end, we specify the fields of the respective collections that are used to link the documents. `localField` defines the field to be used in the arXiv collection and `foreignField` defines the field to be used in the DBLP collection. In other words, each document of the arXiv collection is extended with a new key "`arxivdblp`" that is associated with a list of all DBLP documents that happen to have the same title. To conclude, MongoDB returns documents that appear in both collections with the exact same title.

The last query, **Q4**, is basically query **Q2** with an additional operation: We put the `explain()` operation [38] after the operation we actually want to execute (in this case, the `find()` operation). Like the `EXPLAIN` keyword in PostgreSQL, this instructs MongoDB to return the **query plan**, that is, information about the steps MongoDB plans to execute in order to determine the result. As an exercise (not part of this assignment), the other queries could also be extended with a trailing `explain()` operation.

2.6 Access the Data Using Python3

Although the `mongosh` command-line tool can be used to execute queries, a database system is typically accessed by an application. Therefore, the fourth part of this assignment is to write a small Python3 application. We recommend to use the `pymongo` module (or driver) for Python3 to (a) **establish** a **connection** to your local database, (b) **execute** the **queries** and **retrieve** the results, and (c) **close** the **connection** to your local database. Your application should execute the queries **Q1**, **Q2**, and **Q4** as provided in Section 2.5 and print the respective results (i.e., all documents that are returned by MongoDB; output format does not matter as long as it is human-readable).

Like in Assignment 1, you are asked to adapt query **Q3**: **Q3** as given in Section 2.5 returns a list of documents that satisfy the condition in the `$lookup` aggregation operation. In your Python3 application, **Q3** should only return the **number** of documents that satisfy the condition that is expressed using the `$lookup` operation. This can either be accomplish by modifying the MQL statement itself to count the number of documents (**Hint:** Extend the MQL query with the `$count` [39] aggregation operation) or by adapting the Python3 code such that not all the documents are printed but only the number of documents (**Hint:** Use the `len()` function [40]).

Remark: If you cannot execute all queries (for whatever reason), please contact the instructor **before** the submission. We will find a reasonable solution together.

Template Code There are many tutorials regarding the installation [41] and the usage of `pymongo` [42] in combination with MongoDB. Nonetheless, we provide a minimum template code in Python3 that can be used as a starting point. Unfortunately, PDF viewers tend to represent formatted code differently that cause troubles when copying the code from the PDF into another file. Therefore, we do **not** include the template code in this assignment description but as a separate `.py` file that is available via the course website [43].

2.7 Questionnaire

The questionnaire contains questions about this assignment. These questions are potentially discussed during the after-assignment meetings. The questionnaire can be found in a separate

[38] The `explain` operation: `https://docs.mongodb.com/manual/reference/method/db.collection.explain/`
[39] The `$count` operation: `https://docs.mongodb.com/manual/reference/operator/aggregation/count/`
[40] Python's `len()` function: `https://docs.python.org/3/library/functions.html#len`
[41] Installation of the `pymongo` module: `https://pymongo.readthedocs.io/en/stable/installation.html` and `https://pypi.org/project/pymongo/`
[42] `pymongo` tutorial: `https://pymongo.readthedocs.io/en/stable/tutorial.html`
[43] Template code: `https://dbresearch.uni-salzburg.at/teaching/2024ss/dim/assignment2-template.py`

text file named `assignment2-questionnaire.txt`.

3 Submission

Please submit a single compressed archive (e.g., `.zip` or `.tar.gz`) that contains exactly two files: (a) Your Python3 code and (b) your answers to the questionnaire.

Code

Remark: Please consider removing the database credentials (i.e., username and password) before you submit your code (in case you used them).

Please submit a single Python3 file (`.py`) that contains the full code for this assignment, i.e., the connection to the database and the execution of the four queries. The code **must** print the results of **all** four queries **Q1-4** (one after another) when executed **as submitted** (**Hint:** Set up another virtual machine and try whether it runs). We will **not** debug your code, for example, change some variable to make it work. Therefore, please double-check that your Python3 code works as expected and that **all** four queries are executed (recall that query **Q3** needs to be modified as described in Section 2.6).

Questionnaire

Remark: The recommended formats are `.txt` and `.pdf`.

You can answer the questions directly in the text file `assignment2-questionnaire.txt`. If you prefer to use a different application to answer the questions (e.g., Microsoft Word and the likes), you are welcome to do so. In any case, the submitted file must be in one of the following formats: `.txt`, `.pdf`, `.odt`, `.doc`, or `.docx`.

4 Supplementary Material

This section provides a list of pointers to material that may be helpful to solve the assignment.

- MongoDB: `https://www.mongodb.com/`
- The full MongoDB documentation: `https://docs.mongodb.com/manual/`
- MongoDB introduction: `https://docs.mongodb.com/manual/introduction/`
- MongoDB "Getting Started": `https://docs.mongodb.com/manual/tutorial/getting-started/`
- MongoDB resources regarding the installation of MongoDB on
  - Linux systems: `https://docs.mongodb.com/manual/administration/install-on-linux/`
  - MacOS systems: `https://docs.mongodb.com/manual/tutorial/install-mongodb-on-os-x/`
  - Windows systems: `https://docs.mongodb.com/manual/tutorial/install-mongodb-on-windows/`
- Other resources regarding installation and usage of MongoDB:
  - Two of the many MongoDB "Getting Started" guides:
    1. Windows: `https://www.youtube.com/watch?v=RsWdj7V2Ojg`
    2. Linux/MacOS: `https://www.youtube.com/watch?v=bKjH8WhSu_E`
  - SQL to MongoDB Mapping Chart: `https://docs.mongodb.com/manual/reference/sql-comparison/`
- MongoDB documents: `https://docs.mongodb.com/manual/core/document/`
- MongoDB collections: `https://docs.mongodb.com/manual/core/databases-and-collections/`
- Reference for the `mongosh` command-line tool: `https://www.mongodb.com/docs/mongodb-shell/`

- Python Modules: `https://docs.python.org/3/installing/index.html`
- The pymongo module: `https://pymongo.readthedocs.io/en/stable/`
- Installation of the pymongo module for Python:
  - Official website: `https://pymongo.readthedocs.io/en/stable/installation.html`
  - The Python Package Index: `https://pypi.org/project/pymongo/`
- The pymongo tutorial: `https://pymongo.readthedocs.io/en/stable/tutorial.html`
- One of the many pymongo "Getting Started" guides: `https://www.youtube.com/watch?v=rE_bJl2GAY8`
- One of the many introductions to the Linux terminal: `https://www.digitalocean.com/community/tutorials/an-introduction-to-the-linux-terminal`

5 Grading

For the sake of transparency, this section provides more details on the grading of this assignment, i.e., which part contributes how many points to the total number of 20 points.

Code The code contributes at most 10 points and is evaluated based on the following criteria (if the code is executed as submitted; disregarding the credentials):

Max. Points	Criterion
1.5	*Q1 is executed and the correct result is printed to the command* line.
1.5	*Q2 is executed and the correct result is printed to the command* line.
1.5	*Modified Q3 is executed and the correct result is printed to the command line.*
1.5	*Q4 is executed and the correct result is printed to the command* line.
2+2	*Answer two questions on your code in the after-assignment meeting correctly.*
10	

Questionnaire The questionnaire contributes at most 10 points and is evaluated based on the following criteria (taking the discussion in the after-assignment into account):

Max. Points	Criterion
1.5	*Correctness of answer A1.*
1.5	*Correctness of answer A2.*
1.5	*Correctness of answer A3.*
1.5	*Correctness of answer A4.*
2+2	*Answer two additional questions in the after-assignment meeting correctly.*
10	