School of Information Technology
Department of Computer Science

UNIVERSITEIT VAN PRETORIA
UNIVERSITY OF PRETORIA
YUNIBESITHI YA PRETORIA
Faculty of Engineering, Built Environment and Information Technology

COS326 Database Systems:
Practical 7 2024

Release Date: 18 September 2024
Submission Date: 06 October 2024 @ 23:59
Lecturer: Mr S.M Makura

Total: 50 Marks

Objectives

1. Get exposure to the MongoDB document DBMS.
2. Learn how to create and use documents, collections and JavaScript functions for a MongoDB database using MongoDB Shell.
3. Use aggregation pipelines for advanced queries to analyse hospital data.
4. Appreciate the differences between SQL and NoSQL databases.

For this practical exercise you will use the MongoDB database. When you are done: You must submit the following files:

1. You must submit files, named:
   a. ***.mongoshrc.js*** with all the JavaScript functions for Task 1 and Task 2.
   b. Compress the above documents into an archive and upload it to ClickUP **before** the due date/time. The file name for the archive must have your student number as part of the file name,
   e.g. **uxxxxxxxx-prac7.zip** or **uxxxxxxxx-prac7.tar.gz** where **xxxxxxxx** is your student number.
2. The practical will be marked through a live demo on Discord.

NO LATE submissions will be accepted after the submission date and time has lapsed. Do not wait till the last minute to submit and start giving excuses that you faced technical challenges when you tried to submit.

Task 1: BASIC CRUD OPERATIONS [20 marks]

Scenario

You have been tasked with designing a MongoDB-based Hospital Management System to manage information about patients, doctors, and their appointments. You will need to implement various functions to manage the system's data, including queries and updates for patient records and appointment statistics.

Create the JavaScript functions given in the table below and store them in the file. *mongoshrc.js* in your home folder.

Mongo shell function name	**Description**
`insertPatients(dbName, colName, patientCount)`	Inserts documents representing patients into a collection in the format: `{ "name":"Palesa Mohlare, "age": 34, "ailment": "Cold", "doctor": "Dr. Green", "admitted": true, "appointments": [ { "date": "2024-09-01", "reason": "Cold checkup"}, { "date": "2024-09-15", "reason": "Follow-up"} ] }` Insert `patientCount` number of patients with random names, ages, and ailments.
`findAdmittedPatients(dbName, colName)`	Find and return all patients who are currently admitted (i.e., where "admitted" is "true").
`updatePatientAdmission(dbName, colName, patientName, status)`	Update the admission status of a patient with the given "patientName" by setting "admitted" to the specified "status" (true/false).

`removeDischargedPatients(dbName, colName)`	Remove all patients who have been discharged (i.e. where "admitted" is "false").

Ensure that you have downloaded and installed MongoDB Shell. You can download it from here: https://www.mongodb.com/try/download/shell

Test the above functions in MongoDB Shell as follows:

1. Create a database called "**HospitalDB**" and a collection called "**Patients**".
2. Insert 10 random patients into the "**Patients**" collection using `insertPatients("HospitalDB", "Patients", 10)` . (5 Marks)
3. Run `findAdmittedPatients("HospitalDB", "Patients")` to list all admitted patients. (5 Marks)
4. Update the admission status of a patient (e.g., ""Palesa Mohlare"") using `updatePatientAdmission("HospitalDB", "Patients", " Palesa Mohlare ", true)`. (5 Marks)
5. Remove discharged patients using `removeDischargedPatients("HospitalDB", "Patients")`.(5 Marks)

Task 2: ADVANCED AGGREGATION QUERIES [30 marks]

In this task, you will manage doctors, appointments, and patients using advanced MongoDB queries, including aggregation pipelines to gather statistics about doctors and their appointments.

1. `doctorStats(dbName, colName)`

Use an aggregation pipeline to list all doctors and the number of patients they are currently treating, sorted in ascending order by the doctor's name. (8 marks)

2. `doctorPatientList(dbName, colName, doctorName)`

Query the list of patients under a specific doctor, showing all the patients the doctor is currently treating. (8 marks)

3. `activeDoctorsMR(dbName, colName)`

Use an aggregation pipeline to count the total number of patients for each doctor and store the results in a new collection called "DoctorActivity". (7 marks)

4. `appointmentStats(dbName, colName)`

Use an aggregation pipeline to display the total number of appointments each doctor has, grouped by doctor name. The appointments are stored in a new collection called "Appointments" in the format:

```
{
    "doctor":"Doctor Name",
    "patient":"Patient Name",
    "date":"Appointment Date"
}
```

(7 marks)

Test the above functions in MongoDB Shell as follows:

1. Use `doctorStats("HospitalDB", "Patients")` to show the total number of patients each doctor is treating.
2. Query the list of patients under a specific doctor (e.g Dr. Mandela) using `doctorPatientList("HospitalDB", "Patients", "Dr. Mandela")`
3. Use `activeDoctorsMR("HospitalDB", "Patients")` to compute and store the number of patients for each doctor.
4. Use `appointmentStats("HospitalDB", "Patients")` to display the total number of appointments each doctor has.