Data Science

Lab. Sheet 8 – NoSQL

Introduction

In this lab you will explore Document based and Graph based NoSQL databases using Mongodb and Neo4J.

Exercise 1 (Document Based MongoDB)

Access a free online MongoDB terminal at the following link

https://www.jdoodle.com/online-mongodb-terminal

Introduction

MongoDB stores the data in the form of JSON documents. A group of such documentation is collectively known as a collection in MongoDB. Thus, a collection is analogous to a table in a relational database while a document is analogous to a record.

To store documents, we first need to create a collection. The exciting thing about a NoSQL database is that unlike SQL database, you need not specify the column names or data types in it.

The first step towards creating a collection is to create a database. To create a database and connect to it using the command line, execute the below

```
>use myDB
```

To create a collection, execute the below command:

```
> db.createCollection('firstCollection')
```

As discussed above, it is possible to insert almost any JSON document into every MongoDB collection.

Let us start with insertion of the first JSON document into the firstCollection collection created above.

```
> db.firstCollection.insertOne({name:'Abhishek',skill:'MongoDB'})
```

The above command inserts a single JSON document into the firstCollection. The insertion can be verified by using the command shown below:

```
> db.firstCollection.find();
```

The above command has multiple uses depending on the variation of the find() function. When there are no arguments specified as is the case with the above command, it fetches all the available documents from the collection.

Investigation

Perform the following steps recording the statements used in each step.

a. Create a new collection named inventory and insert the following data using the insertMany() command

```
[{ item: "journal", qty: 25, status: "A",
        size: { h: 14, w: 21, uom: "cm" }, tags: [ "blank", "red" ] },
{ item: "notebook", qty: 50, status: "A",
        size: { h: 8.5, w: 11, uom: "in" }, tags: [ "red", "blank" ] },
{ item: "paper", qty: 100, status: "D",
        size: { h: 8.5, w: 11, uom: "in" }, tags: [ "red", "blank", "plain" ] },
{ item: "planner", qty: 75, status: "D",
        size: { h: 22.85, w: 30, uom: "cm" }, tags: [ "blank", "red" ] },
{ item: "postcard", qty: 45, status: "A",
        size: { h: 10, w: 15.25, uom: "cm" }, tags: [ "blue" ] }]
```

b. Use the getCollectionNames() command to verify the collections in the database and use the find() command to verify that the data has been inserted into the new collection.

c. Use the find() command to list the documents in inventory collection with a status of “D”.

d. Write a query to select all documents where the field size equals the document { h: 14, w: 21, uom: "cm" }

e. Write a query to select all documents where the field uom nested in the size field equals the string value "in".

f. Write a query to select all documents where tags is an array that contains the string "red" as one of its elements.

g. Write a query to select all documents where the field tags value is an array with exactly two elements, "red" and "blank", in the specified order.

h. Explore online how to update documents. Use an appropriate command to update the tag on the postcard item in the inventory collection to also include pink.

i. Explore online how to delete documents in the inventory collection. Delete the paper item.

Exercise 2 (Graph Based Neo4j)

Register to access a free online sandbox for Neo4J at the link below
https://neo4j.com/sandbox-v2/

Introduction

The data model of graph databases is called the labeled Property Graph Model. It is comprised of

- Nodes: The entities in the data.
- Labels: Each node can have one or more label that specifies the type of the node.
- Relationships: Connect two nodes. They have a single direction and type.
- Properties: Key-value pair properties can be stored on both nodes and relationships.

Investigation

Once you register you should access the demo databases through
https://10-0-1-133-34884.neo4jsandbox.com/browser/

Click on Favourites and select the Movie Example Graph to explore. Switch to the database tab to explore the database in detail examining the nodes, relationships and properties.

With the help of the Movie Graph tutorial you should perform the following:

a. Use the Cypher script on Page 2 of the tutorial to add The Matrix to the graph.
b. Write a query to show The Matrix and expand the graph to view the data relating to the movie. View the data in graph and table form.
c. Use a suitable query to show all the persons. View the data in graph and table form.
d. Use a suitable query to find movies released between 1992 and 1998.
e. Based on the script used to add The Matrix, add your own name as a person in the graph and then set yourself as an actor in The Matrix.
f. Rerun the script in part b) to verify that you have been added as an actor to The Matrix.
g. Write a query to list all movies that Buster Keaton has acted in.
h. Write a query to show all the co-actors with Buster Keaton.
i. Explore the remainder of the tutorial.

Exercise 3 (Key Value Pair Based NoSQL)

Search online for a tutorial which allows you to explore a Key Value based NoSQL database such as CouchDB.

Exercise 4 (Column Based NoSQL)

Search online for a tutorial which allows you to explore a Column based NoSQL database such as BigTable by Google.