NoSQL Assignment-2

WORKBOOK – Migrating SQL Data to NoSQL Database

Question:

You should take information from a relational database and migrate it to a NoSQL database of your own choosing.

1. For the relational database, you might use the flight's database, any applicable database,
2. For the NoSQL database, you may use MongoDB, Neo4j, or another NoSQL database of your choosing.
3. Your migration process needs to be reproducible. R/Python code is encouraged. You should also briefly describe the advantages and disadvantages of storing the data in a relational database vs. your NoSQL database.

Overall Take any use case. Implement the backend using any NoSQL database format. Create a simple interface to show the results. You should conclude your work with pointers included for future work.

Solution:

First, let us create a database called company1 in MySQL.

```
mysql> CREATE DATABASE COMPANY1;
Query OK, 1 row affected (0.01 sec)

mysql> use company1;
Database changed
```

Then, create a table called employee corresponding to our use-case.

```
mysql> CREATE TABLE employee (
    -> emp_id INT PRIMARY KEY,
    -> first_name VARCHAR(40),
    -> last_name VARCHAR(40),
    -> birth_day DATE,
    -> sex VARCHAR(1),
    -> salary INT,
    -> super_id INT,
    -> branch_id INT );
Query OK, 0 rows affected (0.26 sec)
```

Finally, let us populate the table with some relevant records. The table is populated and is shown below:

```
mysql> select * from employee;
+--------+------------+-----------+------------+------+--------+----------+-----------+
| emp_id | first_name | last_name | birth_day  | sex  | salary | super_id | branch_id |
+--------+------------+-----------+------------+------+--------+----------+-----------+
|      1 | John       | Doe       | 1990-01-01 | M    |  50000 |     NULL |         1 |
|      2 | Jane       | Smith     | 1992-05-15 | F    |  45000 |        1 |         2 |
|      3 | Mark       | Johnson   | 1985-09-22 | M    |  70000 |     NULL |         3 |
|      4 | Emily      | Davis     | 1988-11-03 | F    |  60000 |        3 |         2 |
|      5 | Michael    | Lee       | 1993-03-10 | M    |  55000 |        4 |         1 |
|      6 | Samantha   | Wilson    | 1987-07-18 | F    |  65000 |        1 |         3 |
|      7 | David      | Brown     | 1984-02-28 | M    |  80000 |     NULL |         2 |
|      8 | Rachel     | Taylor    | 1991-12-20 | F    |  55000 |        7 |         3 |
|      9 | Brandon    | Thomas    | 1990-04-14 | M    |  60000 |        7 |         1 |
|     10 | Jessica    | Hernandez | 1986-06-30 | F    |  75000 |        3 |         2 |
|     11 | Daniel     | Garcia    | 1992-09-01 | M    |  40000 |       10 |         1 |
|     12 | Avery      | Nguyen    | 1994-12-25 | F    |  55000 |        4 |         3 |
|     13 | Ethan      | Chen      | 1989-08-08 | M    |  65000 |        6 |         2 |
|     14 | Olivia     | Miller    | 1991-01-12 | F    |  70000 |        3 |         1 |
|     15 | Samuel     | Taylor    | 1985-05-05 | M    |  90000 |        7 |         2 |
|     16 | Madison    | Clark     | 1994-07-22 | F    |  55000 |        5 |         1 |
|     17 | William    | Wilson    | 1986-11-17 | M    |  60000 |        4 |         3 |
|     18 | Isabella   | Brown     | 1993-03-31 | F    |  75000 |        7 |         2 |
|     19 | Noah       | Garcia    | 1988-09-07 | M    |  40000 |       10 |         1 |
|     20 | Sophia     | Nguyen    | 1992-12-14 | F    |  55000 |        4 |         3 |
+--------+------------+-----------+------------+------+--------+----------+-----------+
20 rows in set (0.00 sec)
```

Migration of data from MySQL to NoSQL (using MongoDB)

Step 1: Installing the required modules

In [12]:
```
#Step 1: Install the Required Modules
pip install mysql-connector-python
pip install pymongo
```

- **mysql-connector-python** is a Python driver for MySQL. It allows Python programs to connect to MySQL databases, execute SQL queries, and manage transactions. The driver is written in pure Python and is compatible with Python 2.7 and Python 3.x.
- **pymongo** is a Python library that provides a driver for MongoDB, a popular document-oriented NoSQL database. pymongo allows Python programs to connect to MongoDB databases, perform CRUD (create, read, update, delete) operations, and perform advanced queries and aggregation tasks. pymongo is built on top of the bson library, which provides a way to encode and decode BSON (Binary JSON) data.

Step 2: Importing the modules

In [1]:
```
#Step 2: Importing the installed modules
import mysql.connector
import pymongo
```

Step 3: Connect to MySQL

In [2]:
```
#Step 3: Connect to MySQL
mysql_conn = mysql.connector.connect(
 host="localhost",
 user="root",
 password="*********",
 database="company1"
)
```

This creates a connection to a MySQL database called "company1" running on the same machine as the Python script, using the username "root" and the appropriate password.

Step 4: Connect to MongoDB

In [3]:
```
#Step 4: Connect to MongoDB
mongodb_host = "mongodb://localhost:27017/"
mongodb_dbname = "mymongodb"
myclient = pymongo.MongoClient(mongodb_host)

mydb = myclient["mydatabase"]
mycol = mydb["mycollection"]


```

This creates a connection to a MongoDB database called "mydatabase" running on the same machine as the Python script, and selects a collection called "mycollection".

Step 5: Read the data from MySQL Table

In [4]:
```
#Step 5: Read the Data from MySQL Table
mysql_cursor = mysql_conn.cursor()
mysql_cursor.execute("SELECT * FROM employee")
data = mysql_cursor.fetchall()
```

This creates a cursor object to interact with the MySQL database, executes a SQL query to select all rows from a table called "employee", and fetches all the resulting data.

Step 6: Transforming the data to MongoDB Format

In [5]:
```
#Step 6: Transform the Data to MongoDB Format
mongo_data = []
for row in data:
    doc = {
    "emp_id": row[0],
    "first_name": row[1],
    "last_name": row[2],
    #"birth_day": row[3],
    "sex":row[4],
    "salary_id":row[5],
    "super_id":row[6],
    "branch_id":row[7]
    }
    mongo_data.append(doc)

```

This iterates over each row of data, creates a new dictionary object for each row, and assigns each value to a corresponding field in the dictionary. The resulting list of dictionaries is stored in the `mongo_data` variable.

Step 7: Insert the data into MongoDB

```
In [6]: 1 #Step 7: Insert the data into MongoDB
        2 mycol.insert_many(mongo_data)
```

```
Out[6]: <pymongo.results.InsertManyResult at 0x1871c0bcfd0>
```

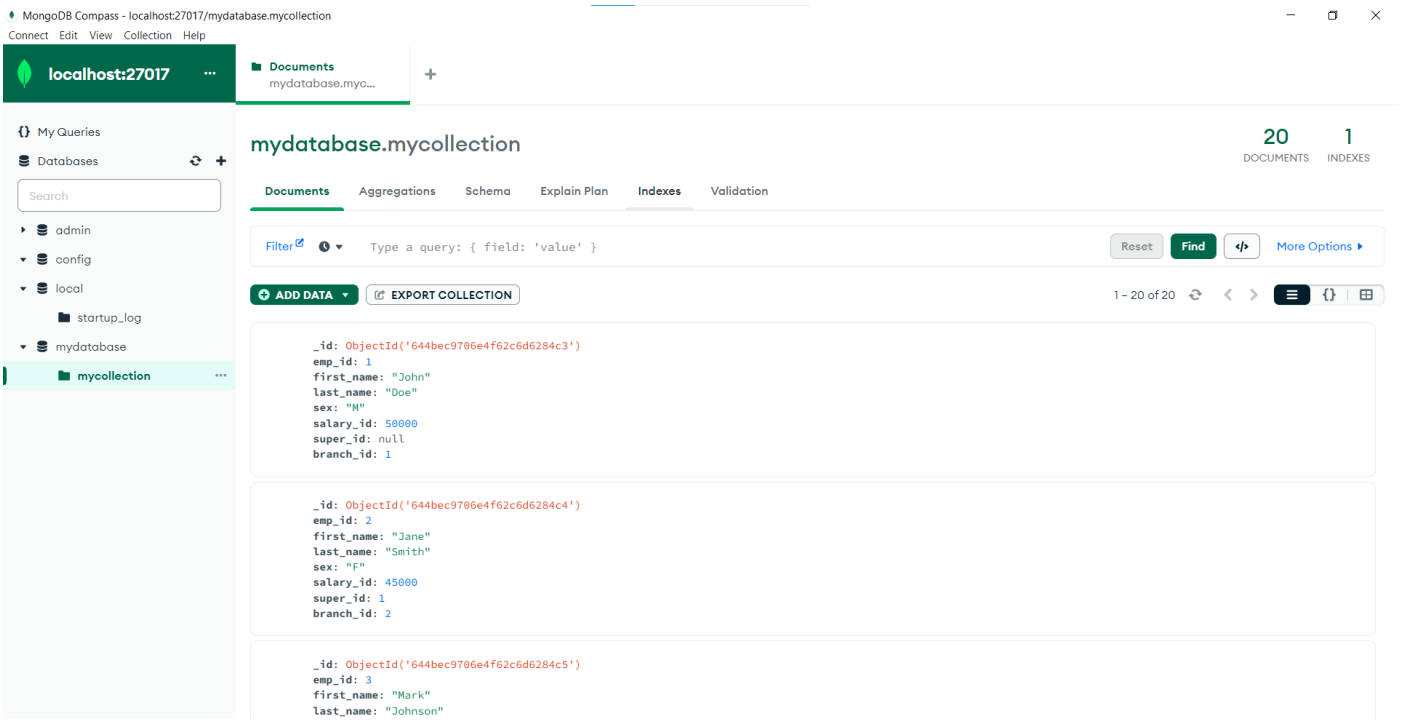
This inserts the list of dictionaries from `mongo_data` into the MongoDB collection specified by `mycol`.

Step 8: Close all the connections

```
In [7]: 1 #Step 8: Close the connections
        2 mysql_cursor.close()
        3 mysql_conn.close()
        4 myclient.close()
        5
```

This closes the connections to both the MySQL and MongoDB databases.

Let us use MongoDB Compass installed to verify if the migration was successfully done.



As we can observe, all the 20 rows from the MySQL employee table have successfully been converted into corresponding NoSQL documents.

Advantages of NoSQL Databases:

- Flexible Schema: NoSQL databases like MongoDB allow for flexible schema design, which can accommodate a wide range of data structures and formats.
- High Scalability: NoSQL databases are highly scalable and can handle large amounts of data without compromising performance.
- High Availability: NoSQL databases can be designed to provide high availability and fault tolerance, which is important for mission-critical applications.

Disadvantages of NoSQL Databases:

- No ACID Transactions: NoSQL databases typically sacrifice ACID transactions for scalability and performance. This means that data consistency and integrity may not always be guaranteed.
- Lack of Standards: NoSQL databases lack a standard query language like SQL, which can make it more difficult to query and analyze data.
- Complexity: NoSQL databases can be more complex to set up and manage than relational databases, especially in distributed environments.

Future work:

There are several potential areas of improvement that could make the migration of data from MySQL to NoSQL databases easier:

1. **Standardized Migration Tools:** Developing standardized migration tools that can handle different types of data and migrate them from MySQL to NoSQL databases would make the process more seamless and reduce the need for custom scripts.
2. **Improved Data Modelling:** NoSQL databases use a different data modelling approach compared to relational databases like MySQL. Therefore, developing better data modelling techniques that can automatically convert data from relational to NoSQL models could make the migration process more straightforward.
3. **Better Documentation and Resources:** Providing better documentation and resources on the differences between relational and NoSQL databases, data modelling techniques, and migration strategies can help users make informed decisions and better plan for the migration process.
4. **Automated Schema Conversion:** Automated schema conversion tools can help convert MySQL schema definitions to NoSQL schema definitions, which can help streamline the migration process.
5. **Performance Testing:** Performance testing tools can help users understand how their application will perform when migrated to a NoSQL database. This can help identify performance bottlenecks and optimization opportunities before migration.

Names: **Mohammed Zakriah Ibrahim (1CR19CS095)**

Mukesh Kumar (1CR19CS099)

Nestor Jameson (1CR19CS204)