17th Feb Assignment

February 27, 2023

1 Assignment 16

Q1. What is MongoDB? Explain non-relational databases in short. In which scenarios it is preferred to use MongoDB over SQL databases?

Ans.MongoDB is a popular document-oriented NoSQL database that uses JSON-like documents with dynamic schemas. It is designed to provide high performance, high availability, and automatic scaling.

Non-relational databases, also known as NoSQL databases, are databases that do not use the traditional tabular relations used by relational databases such as SQL. Instead, they store data in a variety of formats such as key-value pairs, document-based, column-based, and graph-based formats. NoSQL databases are designed to be highly scalable and flexible, making them a good choice for handling large, unstructured data sets.

MongoDB is often preferred over SQL databases in scenarios where the data being stored is unstructured or semi-structured. This includes data such as social media feeds, sensor data, and log data. MongoDB's flexible document-based data model makes it easy to store and retrieve data without having to define a schema upfront. Additionally, MongoDB's ability to shard data across multiple servers makes it a good choice for handling large amounts of data and scaling up as needed.

Q2. State and Explain the features of MongoDB.

Ans.MongoDB is a popular NoSQL document-oriented database that provides various features that make it a preferred choice in many scenarios. Some of its features are:

- Document-oriented: MongoDB is a document-oriented database, which means that it stores data in the form of documents, similar to JSON format. This makes it easy to store complex hierarchical data structures and flexible data models.
- Schema-less: MongoDB is schema-less, meaning that it does not enforce a fixed data model or schema. This makes it easier to store and retrieve data with different structures.
- Distributed: MongoDB is designed to work in a distributed environment, which means that it can be easily scaled across multiple servers to handle large amounts of data and high traffic.
- High availability: MongoDB supports replica sets, which provide high availability and automatic failover in case of a primary node failure.

- Indexing: MongoDB provides various indexing options, including compound indexes and geospatial indexes, to improve query performance.
- Aggregation: MongoDB provides a powerful aggregation framework that allows for complex data analysis and manipulation

Q3. Write a code to connect MongoDB to Python. Also, create a database and a collection in MongoDB.

Ans.To connect MongoDB to Python, we first need to install the pymongo library. You can install it using the following command

[1]:
```
pip install pymongo
```

```
Collecting pymongo
  Downloading
pymongo-4.3.3-cp310-cp310-manylinux_2_17_x86_64.manylinux2014_x86_64.whl (492
kB)
                            492.9/492.9

kB 6.4 MB/s eta 0:00:0000:0100:01
Collecting dnspython<3.0.0,>=1.16.0
  Downloading dnspython-2.3.0-py3-none-any.whl (283 kB)
                            283.7/283.7 kB
35.2 MB/s eta 0:00:00
Installing collected packages: dnspython, pymongo
Successfully installed dnspython-2.3.0 pymongo-4.3.3
Note: you may need to restart the kernel to use updated packages.
```

[2]:
```
import pymongo

# Establishing a connection to MongoDB
client = pymongo.MongoClient("mongodb://localhost:27017/")

# Creating a database
mydb = client["mydatabase"]

# Creating a collection
mycol = mydb["customers"]
```

In the above code, we first import the pymongo library. Then we establish a connection to MongoDB using the MongoClient() function. The connection URL specifies the host and port number of the MongoDB server.

Q4. Using the database and the collection created in question number 3, write a code to insert one record,and insert many records. Use the find() and find_one() methods to print the inserted record.

[ ]:
```
import pymongo

# Connect to MongoDB server
client = pymongo.MongoClient("mongodb://localhost:27017/")

# Create a database and a collection
mydb = client["mydatabase"]
mycol = mydb["customers"]

# Insert one record
record = {"name": "John", "address": "Highway 37"}
x = mycol.insert_one(record)

# Print the inserted record
print("Inserted one record with ID:", x.inserted_id)
print("Inserted record:")
print(mycol.find_one())

# Insert many records
records = [
    {"name": "Amy", "address": "Apple st 652"},
    {"name": "Hannah", "address": "Mountain 21"},
    {"name": "Michael", "address": "Valley 345"},
]
x = mycol.insert_many(records)

# Print the inserted records
print("Inserted many records with IDs:", x.inserted_ids)
print("Inserted records:")
for record in mycol.find():
    print(record)
```

Q5. Explain how you can use the find() method to query the MongoDB database. Write a simple code to demonstrate this.

Ans.The find() method is used in MongoDB to query the database and retrieve documents that match certain criteria. It takes a query object as its argument and returns a cursor object that can be used to iterate through the matched documents.

[ ]:
```
import pymongo

# establish a connection to the MongoDB server
client = pymongo.MongoClient("mongodb://localhost:27017/")

# create a database and a collection
mydb = client["mydatabase"]
mycol = mydb["customers"]
```

```python
# insert some documents into the collection
mycol.insert_many([
    {"name": "John", "address": "Highway 37"},
    {"name": "Jane", "address": "Baker Street"},
    {"name": "Bob", "address": "Main Street"}
])

# query the collection for documents with the name "John"
query = {"name": "John"}
results = mycol.find(query)

# iterate through the matched documents and print them
for result in results:
    print(result)
```

Q6. Explain the sort() method. Give an example to demonstrate sorting in MongoDB.

Ans.In MongoDB, delete_one() method is used to delete the first document that matches the specified condition from a collection, while delete_many() method is used to delete all the documents that match the specified condition. Both methods return a DeleteResult object, which contains information about the operation such as the number of documents that were deleted.

The drop() method is used to delete an entire collection from a database. This method does not take any parameters, and simply deletes the entire collection. It is a more drastic method of deleting data and should be used with caution. Once a collection is dropped, all the data it contained will be lost permanently.

These methods are used to delete unwanted data from collections or entire collections itself.