AROR UNIVERSITY
OF ART, ARCHITECTURE,
DESIGN & HERITAGE,
SUKKUR, SINDH

Department of Artificial Intelligence & Multimedia Gamming

CSC-207: Database Systems

Lab # 12: Introduction of NoSQL

Objectives

- Understand the basic concepts of NoSQL databases.
- Explore the different types of NoSQL databases.
- Understand Document-oriented database (MongoDB).
- Understand how to install MongoDB.
- Understand the basic CRUD operations in MongoDB.

Introduction to NoSQL Databases

- NoSQL stands for "Not Only SQL".
- It is a type of database that is designed to handle large volumes of unstructured or semi-structured data.
- NoSQL databases are often used in big data and real-time web applications.
- NoSQL databases are designed to be highly scalable, flexible, and performant. They often sacrifice some of the strict consistency guarantees of traditional SQL databases for improved performance and scalability.
- NoSQL databases use various data structures, including JSON-like documents, key-value pairs, graph databases, and column-family stores.

Types of NoSQL Databases

There are several types of NoSQL databases, including:

- **Key-Value Stores:** These databases store data as key-value pairs, where each key is associated with a value. Examples include Redis and Amazon DynamoDB.
- **Document Databases:** These databases store data as documents, which are typically JSON-like structures. Examples include MongoDB and Couchbase.

- **Column-Family Stores:** These databases store data in column families, which are groups of related columns. Examples include Apache Cassandra and HBase.
- **Graph Databases:** These databases store data as nodes and edges, which represent entities and relationships between them. Examples include Neo4j.

Document-oriented database (MongoDB)

- MongoDB is a popular document-oriented NoSQL database.
- It stores data in flexible, JSON-like documents.
- MongoDB is known for its scalability, performance, and ease of use.
- MongoDB is a cross-platform document-oriented database program. Classified as a NoSQL database program, MongoDB uses JSON-like documents with optional schemas. MongoDB is developed by MongoDB Inc. and licensed under the Server Side Public License (SSPL).

Installing MongoDB

- Install MongoDB on your system following the installation instructions for your operating system.
- Once installed, you can start the MongoDB server using the mongod command.

Basic CRUD Operations in MongoDB

- **CRUD** stands for **Create**, **Read**, **Update**, and **Delete**. These are the four basic operations that can be performed on data in a database.
    - **Create:** Insert new documents into a collection.
    - **Read:** Retrieve documents from a collection.
    - **Update:** Modify existing documents in a collection.
    - **Delete:** Remove documents from a collection.
- In MongoDB, you can perform CRUD operations using the mongo shell or a MongoDB driver for your programming language.
- **Create (Insert):**
    - To insert a single document, you can use the insertOne() method.
    - To insert multiple documents, you can use the insertMany() method.

```
// Insert a single document
db.collection.insertOne({ name: "John Doe", age: 30 });
// Insert multiple documents
db.collection.insertMany([
  { name: "Jane Smith", age: 25 },
  { name: "Bob Johnson", age: 40 }
```

```
]);
```

- **Read (Find):**
  - To retrieve documents from a collection, you can use the find() method.
  - The find() method returns a cursor, which you can iterate over to access the documents.
  - You can also use the findOne() method to retrieve a single document that matches the query.

```
// Find all documents in the collection
db.collection.find();

// Find documents that match a query
db.collection.find({ age: { $gte: 30 } });

// Find a single document that matches the query
db.collection.findOne({ name: "John Doe" });
```

- **Update:**
  - To modify existing documents in a collection, you can use the updateOne() or updateMany() method.
  - These methods take two arguments: a query that specifies which documents to update, and an update document that specifies how to modify the documents.

```
// Update a single document
db.collection.updateOne(
 { name: "John Doe" },
 { $set: { age: 31 } }
);

// Update multiple documents
db.collection.updateMany(
 { age: { $gte: 30 } },
 { $inc: { age: 1 } }
);
```

- **Delete:**
  - To remove documents from a collection, you can use the deleteOne() or deleteMany() method.
  - These methods take a query that specifies which documents to remove.

```
// Delete a single document
db.collection.deleteOne({ name: "John Doe" });

// Delete multiple documents
db.collection.deleteMany({ age: { $gte: 30 } });
```

What is MongoDB Atlas?

MongoDB Atlas is a fully managed cloud database service provided by MongoDB. It simplifies the process of deploying, operating, and scaling MongoDB databases in the cloud. Atlas handles the infrastructure, software installation, and maintenance, allowing developers to focus on building applications.

Key features of MongoDB Atlas include:

MongoDB Atlas is a fully managed cloud database service offering automated operational tasks like provisioning, configuration, patching, upgrades, backups, and monitoring. It provides multi-cloud support across AWS, Azure, and Google Cloud, ensuring scalability and high availability.

Signing Up for a MongoDB Atlas Account

To begin using MongoDB Atlas, you'll need to create an account. Follow these steps:

1. Open your web browser and go to the MongoDB Atlas website (cloud.mongodb.com).
2. Click on the "Start Free" or "Sign Up" button.
3. Provide your email address, first name, last name, and password. Alternatively, you can sign up using your Google account.
4. Agree to the Terms of Service and Privacy Policy by checking the appropriate box.
5. Click the "Sign Up" button to create your account.
6. MongoDB Atlas may send a verification email to your registered email address. Follow the instructions in the email to verify your account.
7. Once your account is verified, you can log in to the MongoDB Atlas dashboard.

Creating Your First MongoDB Atlas Cluster

After signing up, your next step is to create your first MongoDB Atlas cluster. A cluster is a set of MongoDB processes that store your data. (Click here)

1. **Log in to the MongoDB Atlas dashboard.**
2. **If you're a new user, you'll be guided through a setup process.** Otherwise, click the "Build a Database" button.
3. **Choose a cluster tier.** Atlas offers different tiers (e.g., Free Tier, Shared Clusters, Dedicated Clusters) with varying resources and features. For learning purposes, the Free Tier is often sufficient.
4. **Select your cloud provider** (AWS, Azure, or Google Cloud) and the region where you want to deploy your cluster. Choose a region that is geographically close to you or your application's users.
5. **Configure your cluster settings.** This may include:
    - **Cluster Name:** Give your cluster a descriptive name.
    - **MongoDB Version:** Select the version of MongoDB you want to use.
    - **Additional settings:** Depending on the tier, you might be able to configure backup options, etc.
6. **Create a database user.** You'll need to create a user with appropriate permissions to access your database. Specify a username and password, and select a role (e.g., readWriteAnyDatabase).
7. **Review your configuration and click the "Create Cluster" button.**
8. **Atlas will begin provisioning your cluster.** This process may take a few minutes. You can monitor the progress in the Atlas dashboard.
9. **Once the cluster is created, you are ready to start working with your database!**

Exercises (Class)

Add here all the tasks performed in lab.

Exercises (Weekly)

1. Install MongoDB on your system.
2. Create a database and a collection.
3. Insert the following documents into a collection named "users":

```
[
 { "name": "Alice", "age": 28, "city": "New York" },
 { "name": "Bob", "age": 35, "city": "Los Angeles" },
 { "name": "Charlie", "age": 22, "city": "New York" },
 { "name": "David", "age": 40, "city": "Chicago" },
 { "name": "Eve", "age": 30, "city": "Los Angeles" }
]
```

4. Find all users in the "users" collection.
5. Find users in the "users" collection who are older than 30.
6. Find users in the "users" collection who live in "New York".
7. Update the age of "Alice" to 29 in the "users" collection.
8. Update the city of all users in "Los Angeles" to "San Francisco".
9. Delete the user named "Charlie" from the "users" collection.
10. Delete all users from the "users" collection who are younger than 25.
11. Create a new collection named "products" and insert the following documents:

```
[
 { "name": "Laptop", "price": 1200, "category": "Electronics" },
 { "name": "T-Shirt", "price": 25, "category": "Clothing" },
 { "name": "Smartphone", "price": 800, "category": "Electronics" },
 { "name": "Jeans", "price": 60, "category": "Clothing" },
 { "name": "Book", "price": 15, "category": "Books" }
]
```

12. Find all products in the "products" collection that belong to the "Electronics" category.