**4.A.Creating Databases using MongoDB, DynamoDB, Voldemort Key-Value Distributed Data Store Hbase and Neo4j.**

**Aim:**

To Create Databases using MongoDB, DynamoDB, Voldemort Key-Value Distributed Data Store Hbase and Neo4j.

**Procedure:**

**MongoDB (Create Database)**

const { MongoClient } = require('mongodb');

// Import the MongoClient class from the 'mongodb' library.

async function createDatabase() {

const uri = "mongodb://127.0.0.1:27017";

// Connection string to connect to the MongoDB server running locally.

const client = new MongoClient(uri);

// Create a new MongoClient instance to connect to the server.

try {

await client.connect();

// Connect to the MongoDB server.

console.log("Connected to MongoDB");

// Log a success message.

const db = client.db("MyMongoDB");

// Create a database named 'MyMongoDB'. If it doesn't exist, MongoDB will create it when data is added.

console.log("Database created: MyMongoDB");

// Log that the database has been created.

} catch (err) {

console.error(err);

// Handle and log any errors that occur during the connection or database creation process.

} finally {

await client.close();

// Close the connection to the MongoDB server to free up resources.

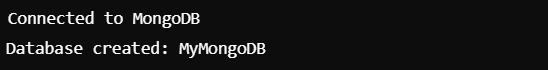
}

}

createDatabase();

// Invoke the function to execute the database creation logic.

**OUTPUT:**

****

**DynamoDB (Create Table)**

const AWS = require('aws-sdk');

// Import the AWS SDK library for interacting with AWS services.

AWS.config.update({

region: "us-east-1",

// Specify the AWS region where DynamoDB is running.

endpoint: "http://localhost:8000"

// Use the local DynamoDB instance endpoint.

});

const dynamoDB = new AWS.DynamoDB();

// Create a DynamoDB service object.

const params = {

TableName: "MyDynamoDBTable",

// Name of the table to be created.

KeySchema: [

{ AttributeName: "id", KeyType: "HASH" }

// Define the primary key schema. 'id' is the partition key.

],

AttributeDefinitions: [

{ AttributeName: "id", AttributeType: "S" }

// Define the data type of the 'id' attribute as a string.

],

ProvisionedThroughput: {

ReadCapacityUnits: 1,

// Set the read capacity units for the table.

WriteCapacityUnits: 1

// Set the write capacity units for the table.

}

};

dynamoDB.createTable(params, (err, data) => {

// Call the `createTable` method to create the table using the defined parameters.

if (err) {

console.error("Error creating table:", JSON.stringify(err, null, 2));

// Log an error message if table creation fails.

} else {

console.log("Table created:", JSON.stringify(data, null, 2));

// Log the success message with table details.

}

});

**OUTPUT**

****

### Voldemort (Create Store)

### **const fetch = require('node-fetch');**

### **// Import the 'node-fetch' library for making HTTP requests.**

### **async function createStore() {**

### **const url = "http://localhost:8080/stores";**

### **// Endpoint for managing Voldemort stores.**

### **const storeConfig = `**

### **<store>**

### **<name>MyVoldemortStore</name>**

### **<!-- Name of the key-value store being created -->**

### **<key-serializer>**

### **<type>string</type>**

### **<!-- Specify that the keys are serialized as strings -->**

### **</key-serializer>**

### **<value-serializer>**

### **<type>string</type>**

### **<!-- Specify that the values are serialized as strings -->**

### **</value-serializer>**

### **</store>`;**

### **try {**

### **const response = await fetch(url, {**

### **method: "POST",**

### **// Use POST method to send the store creation request.**

### **headers: { "Content-Type": "application/xml" },**

### **// Specify that the request body is in XML format.**

### **body: storeConfig**

### **// Send the store configuration as the request body.**

### **});**

### **if (response.ok) {**

### **console.log("Store created successfully");**

### **// Log a success message if the store is created.**

### **} else {**

### **console.error("Error creating store:", response.statusText);**

### **// Log the response's error status if store creation fails.**

### **}**

### **} catch (err) {**

### **console.error("Request failed:", err);**

### **// Handle and log network or other errors.**

### **}**

### **}**

### **createStore();**

### **// Invoke the function to create the store.**

### **OUTPUT**

### 

### HBase (Create Table)

### const HBase = require('hbase-client');

### // Import the HBase client library.

### const config = {

### zookeeperHosts: ["localhost"],

### // Define the Zookeeper hosts for HBase.

### zookeeperRoot: "/hbase",

### // Specify the Zookeeper root path for HBase.

### rpcTimeout: 30000,

### // Timeout for RPC (Remote Procedure Call) operations.

### callTimeout: 30000,

### reconnectTimeout: 30000

### // Timeout for reconnecting to the server.

### };

### const client = HBase.create(config);

### // Create a connection to the HBase server with the specified configuration.

### async function createTable() {

### const tableDescriptor = {

### name: "MyHBaseTable",

### // Define the name of the table.

### families: [{ name: "data" }]

### // Specify a column family named 'data'.

### };

### try {

### await client.createTable(tableDescriptor);

### // Call the createTable method with the table descriptor.

### console.log("Table created: MyHBaseTable");

### // Log a success message if the table is created.

### } catch (err) {

### console.error("Error creating table:", err);

### // Log any errors encountered during table creation.

### } finally {

### client.close();

### // Close the HBase client connection.

### }

### }

### createTable();

### // Invoke the function to execute the table creation logic.

### OUTPUT

### 

### Neo4j (Create Database)

const neo4j = require('neo4j-driver');

// Import the Neo4j driver library for Node.js.

const driver = neo4j.driver("bolt://localhost:7687", neo4j.auth.basic("neo4j", "password"));

// Create a driver instance to connect to the Neo4j database. Replace 'password' with your actual password.

const session = driver.session();

// Open a new session to interact with the database.

async function createDatabase() {

try {

await session.run('CREATE (n:Database {name: "MyNeo4jDB"}) RETURN n');

// Execute a Cypher query to create a node with the label 'Database' and a property 'name'.

console.log("Database node created: MyNeo4jDB");

// Log a success message when the node is created.

} catch (err) {

console.error("Error creating database:", err);

// Log any errors encountered during the database creation process.

} finally {

await session.close();

// Close the session.

await driver.close();

// Close the driver to free resources.

}

}

createDatabase();

// Invoke the function to execute the database creation logic.

**OUTPUT:**

****

**Result:**

Thus to Create Databases using MongoDB, DynamoDB, Voldemort Key-Value Distributed Data Store Hbase and Neo4j has be completed successfully

**4.B.Writing simple queries to access databases created using MongoDB, DynamoDB, Voldemort Key-Value Distributed Data Store Hbase and Neo4j.**

**Aim:**

To Write simple queries to access databases created using MongoDB, DynamoDB, Voldemort Key-Value Distributed Data Store Hbase and Neo4j.

**Procedure:**

### Prerequisites

1. **Install Node.js**: Make sure you have Node.js installed. You can download it from [Node.js official website](https://nodejs.org/).
2. **Set Up VS Code**: Download and install [Visual Studio Code](https://code.visualstudio.com/).
3. **Install Extensions**:
   * MongoDB: [MongoDB for VS Code](https://marketplace.visualstudio.com/items?itemName=mongodb.mongodb-vscode)
   * Neo4j: [Neo4j VS Code Extension](https://marketplace.visualstudio.com/items?itemName=neo4j.neo4j-vscode)

### Step-by-Step Setup Instructions

#### 1. **Initialize a Node.js Project**

Open VS Code, create a new folder for your project, and initialize a Node.js project.

# Open terminal in VS Code and run:

npm init -y

This will create a package.json file where your project dependencies are listed.

#### 2. **Install Required Packages**

Install the necessary npm packages for each database.

# MongoDB

npm install mongodb

# DynamoDB (AWS SDK)

npm install aws-sdk

# Voldemort (REST calls using axios)

npm install axios

# HBase (hbase-client)

npm install hbase-client

# Neo4j

npm install neo4j-driver

#### 3. **Configure JavaScript Code for Each Database**

For each database, create a separate JavaScript file and add the respective code for operations.

### MongoDB Setup (Using mongodb package)

Create a file named mongodb.js and add the following code:

const { MongoClient } = require('mongodb');

const uri = "mongodb://localhost:27017"; // MongoDB connection string

const client = new MongoClient(uri);

async function run() {

try {

await client.connect();

const database = client.db("testDB");

const students = database.collection("students");

// Insert Data

await students.insertOne({ name: "Alice", age: 22, major: "Computer Science" });

// Retrieve All Documents

const allStudents = await students.find({}).toArray();

console.log("All Students:", allStudents);

// Retrieve Specific Documents

const specificStudent = await students.find({ age: 22 }).toArray();

console.log("Specific Student:", specificStudent);

// Update a Document

await students.updateOne({ name: "Alice" }, { $set: { major: "Mathematics" } });

// Delete a Document

await students.deleteOne({ name: "Alice" });

} finally {

await client.close();

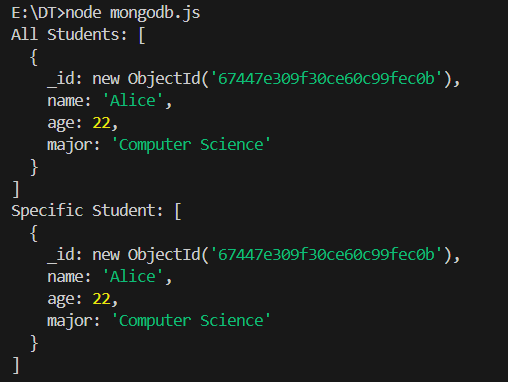
}

}

run().catch(console.dir);

To run this file, open the terminal and execute:  
  
node mongodb.js

**Output:**



### DynamoDB Setup (Using aws-sdk)

Create a file named dynamodb.js and add the following code:

const AWS = require('aws-sdk');

// Configure DynamoDB Local

AWS.config.update({

region: "us-west-2",

endpoint: "http://localhost:8000"

});

const dynamodb = new AWS.DynamoDB.DocumentClient();

const tableName = "Students";

async function run() {

// Insert Data

await dynamodb.put({

TableName: tableName,

Item: { StudentID: "123", Name: "Alice", Age: 22, Major: "Computer Science" }

}).promise();

// Retrieve a Specific Item

const item = await dynamodb.get({

TableName: tableName,

Key: { StudentID: "123" }

}).promise();

console.log("Retrieved Item:", item);

// Update an Item

await dynamodb.update({

TableName: tableName,

Key: { StudentID: "123" },

UpdateExpression: "SET Major = :m",

ExpressionAttributeValues: { ":m": "Mathematics" }

}).promise();

// Delete an Item

await dynamodb.delete({

TableName: tableName,

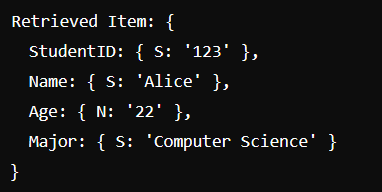
Key: { StudentID: "123" }

}).promise();

}

run().catch(console.error);

Run it using:  
  
node dynamodb.js  
  
**Output:**



### Voldemort Setup (Using axios)

Create a file named voldemort.js and add the following code:

const axios = require('axios');

const baseUrl = "http://localhost:8081/stores/students/";

async function run() {

// Insert Data

await axios.post(baseUrl, {

key: "123",

value: { name: "Alice", age: 22, major: "Computer Science" }

});

// Retrieve Data

const { data } = await axios.get(`${baseUrl}123`);

console.log("Retrieved Data:", data);

// Delete Data

await axios.delete(`${baseUrl}123`);

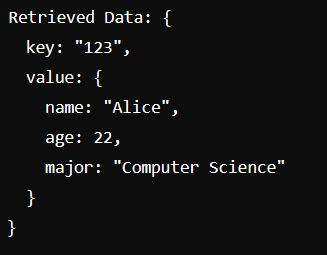
}

run().catch(console.error);

Run it with:

node voldemort.js

**Output:**



### HBase Setup (Using hbase-client)

Create a file named hbase.js and add the following code:

const HBase = require('hbase-client');

const client = HBase.create({

zookeeperHosts: ['localhost:2181'],

zookeeperRoot: '/hbase'

});

async function run() {

// Insert Data

await client.put('students', '1', [

{ column: 'info:name', $: 'Alice' },

{ column: 'info:age', $: '22' },

{ column: 'info:major', $: 'Computer Science' }

]);

// Retrieve Data for a Row

const result = await client.getRow('students', '1');

console.log("Row Data:", result);

// Delete Data

await client.delete('students', '1', 'info:major');

client.close();

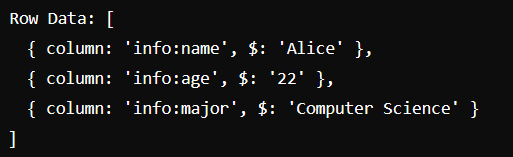
}

run().catch(console.error);

Run with:

node hbase.js

**Output:**



### Neo4j Setup (Using neo4j-driver)

Create a file named neo4j.js and add the following code:

const neo4j = require('neo4j-driver');

const uri = 'bolt://localhost:7687';

const user = 'neo4j';

const password = 'your\_password';

const driver = neo4j.driver(uri, neo4j.auth.basic(user, password));

const session = driver.session();

async function run() {

// Create a Node

await session.run("CREATE (a:Student {name: 'Alice', age: 22, major: 'Computer Science'})");

// Retrieve All Nodes

const result = await session.run("MATCH (s:Student) RETURN s");

result.records.forEach(record => {

console.log("Retrieved Node:", record.get('s').properties);

});

// Update a Node

await session.run("MATCH (s:Student {name: 'Alice'}) SET s.major = 'Mathematics'");

// Delete a Node

await session.run("MATCH (s:Student {name: 'Alice'}) DELETE s");

await session.close();

driver.close();

}

run().catch(console.error);

Run with:

node neo4j.js

**Output:**

****

**Result:**

Thus to Write simple queries to access databases created using MongoDB, DynamoDB, Voldemort Key-Value Distributed Data Store Hbase and Neo4j has been completed successfully.