**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 {

console.log("Attempting to connect to MongoDB...");

// Log the connection attempt.

await client.connect();

// Connect to the MongoDB server.

console.log("Successfully connected to MongoDB!");

// Log a success message after connecting.

const dbName = "MyMongoDB";

// Name of the database to create.

const db = client.db(dbName);

// Create a database with the specified name.

// The database is created only when you add collections or documents to it.

console.log(`Database connection established: ${dbName}`);

// Example: Creating a collection (optional, creates a collection named 'TestCollection').

const collection = db.collection('TestCollection');

await collection.insertOne({ message: "Hello MongoDB!" });

// Insert a test document to ensure the database and collection are operational.

console.log(`Sample collection 'TestCollection' created, and data inserted successfully.`);

} catch (err) {

console.error("An error occurred:", 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.

console.log("Connection to MongoDB closed.");

}

}createDatabase();

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

**Output:**

Attempting to connect to MongoDB...

Successfully connected to MongoDB!

Database connection established: MyMongoDB

Sample collection 'TestCollection' created, and data inserted successfully.

Connection to MongoDB closed.

**DynamoDB (Create Table)**

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

AWS.config.update({

region: "us-east-1",

endpoint: "http://localhost:8000"

});

const dynamoDB = new AWS.DynamoDB();

const params = {

TableName: "MyDynamoDBTable",

KeySchema: [{ AttributeName: "id", KeyType: "HASH" }],

AttributeDefinitions: [{ AttributeName: "id", AttributeType: "S" }],

ProvisionedThroughput: {

ReadCapacityUnits: 1,

WriteCapacityUnits: 1

}

};

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

if (err) {

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

} else {

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

}

});

**OUTPUT:**

Table created successfully: {

"TableDescription": {

"TableName": "MyDynamoDBTable",

"TableStatus": "ACTIVE",

...

}

}

**Voldemort (Create Store):**

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

async function createStore() {

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

const storeConfig = `

<store>

<name>MyVoldemortStore</name>

<key-serializer><type>string</type></key-serializer>

<value-serializer><type>string</type></value-serializer>

</store>`;

try {

const response = await fetch(url, {

method: "POST",

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

body: storeConfig

});

if (response.ok) {

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

} else {

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

}

} catch (err) {

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

}

}

createStore();

**Output:**

Store created successfully.

**HBase (Create Table):**

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

const config = {

zookeeperHosts: ["localhost"],

zookeeperRoot: "/hbase",

rpcTimeout: 30000,

callTimeout: 30000,

reconnectTimeout: 30000

};

const client = HBase.create(config);

async function createTable() {

const tableDescriptor = {

name: "MyHBaseTable",

families: [{ name: "data" }]

};

try {

await client.createTable(tableDescriptor);

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

} catch (err) {

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

} finally {

client.close();

console.log("HBase connection closed.");

}

}

createTable();

**OUTPUT:**

Table created successfullly: MyHBaseTable

HBase connection closed.

**Neo4j (Create Database)**

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

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

const session = driver.session();

async function createDatabase() {

try {

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

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

} catch (err) {

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

} finally {

await session.close();

await driver.close();

console.log("Neo4j connection closed.");

}

}

createDatabase();

**OUTPUT:**

Database node created successfully: MyNeo4jDB

Neo4j connection closed.

**Result:**

Thus, databases or tables were successfully created using:

1. MongoDB
2. DynamoDB
3. Voldemort Key-Value Distributed Data Store
4. HBase
5. Neo4j

**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:**

db.users.find({ age: { $gt: 25 } });

{

TableName: "Users",

Key: {

"userId": 101

}

}

StoreClient<String, String> client = storeClientFactory.getStoreClient("store\_name");

String value = client.getValue("key1");

System.out.println(value);

get 'users', 'row1’

MATCH (u:User)-[:FRIENDS\_WITH]->(friend:User)

WHERE u.name = "John"

RETURN friend.name;

### **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:**

npm install aws-sdk

[default]

aws\_access\_key\_id = YOUR\_ACCESS\_KEY\_ID

aws\_secret\_access\_key = YOUR\_SECRET\_ACCESS\_KEY

region = YOUR\_REGION

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

// Set AWS region

AWS.config.update({ region: "us-east-1" });

// Create DynamoDB service object

const dynamoDB = new AWS.DynamoDB();

const params = {

TableName: "Users",

KeySchema: [

{ AttributeName: "userId", KeyType: "HASH" } // Partition key

],

AttributeDefinitions: [

{ AttributeName: "userId", AttributeType: "S" } // String type

],

ProvisionedThroughput: {

ReadCapacityUnits: 5,

WriteCapacityUnits: 5

}

};

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

if (err) {

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

} else {

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

}

});

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

const params = {

TableName: "Users",

Item: {

userId: "101",

name: "John Doe",

age: 30,

email: "johndoe@example.com"

}

};

docClient.put(params, (err, data) => {

if (err) {

console.error("Error adding item:", JSON.stringify(err, null, 2));

} else {

console.log("Item added successfully:", JSON.stringify(data, null, 2));

}

});

const queryParams = {

TableName: "Users",

KeyConditionExpression: "userId = :id",

ExpressionAttributeValues: {

":id": "101"

}

};

docClient.query(queryParams, (err, data) => {

if (err) {

console.error("Error querying items:", JSON.stringify(err, null, 2));

} else {

console.log("Query result:", JSON.stringify(data, null, 2));

}

});

const deleteParams = {

TableName: "Users"

};

dynamoDB.deleteTable(deleteParams, (err, data) => {

if (err) {

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

} else {

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

}

});

### **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:**

Retrieved Data :

{

Key : “123”,

Value : {

Name : “Alice”,

Age:22,

Major : “ Computer Science”

}

}

### **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:**

Row Data :[

{ Coloumn : ‘info: name’,$:`Alice’}

{ Coloumn: ‘info : age’, $ `22’},

{ Coloumn: ‘info: major’,$: ‘Computer Science'}

]

### **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:**

**Retrieved Node:**

{

Name: ‘Alice’,

Age: 22,

Major : ‘Computer Science “)

**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.