Exp no:12 Document, Column and Graph Based Data Using NoSQL Database Tools

Aim

To install and utilize three types of NoSQL databases—Document-based (MongoDB), Column-based (Apache Cassandra), and Graph-based (Neo4j)—to create and query data, showcasing their different structures and query approaches.

1. MongoDB – Document-Based Database

Explanation

MongoDB is a NoSQL database that stores data in BSON (binary JSON) format. It is schema-less and ideal for applications with rapidly changing or hierarchical data structures. Data is stored in "collections" (similar to tables), and records are stored as "documents".

Installation Steps

1. Download MongoDB Community Edition:
    https://www.mongodb.com/try/download/community

2. After installation, start the MongoDB server:

```
mongod
```

3. Open MongoDB shell:

```
mongo
```

Data Creation and Query

```
// Switch to or create a database
use company

// Insert a document into the 'employees' collection
db.employees.insertOne({
  name: "Alice",
  department: "HR",
  age: 29,
  skills: ["communication", "recruitment"],
  address: {
    city: "New York",
    zip: "10001"
  }
})

// Query documents from the 'employees' collection
db.employees.find({ department: "HR" })
```

Explanation of Commands

- `use company`: Selects or creates the "company" database.
- `insertOne`: Adds a single document to the collection.
- `find`: Retrieves documents based on the specified filter.

2. Apache Cassandra – Column-Based Database

Explanation

Cassandra is a distributed NoSQL database that uses a wide-column store. It is optimized for fast writes and works well for big data and analytics applications. Data is organized in keyspaces and tables (similar to databases and tables in SQL).

Installation Steps

1. Install Java JDK (version 8 or higher).
2. Download Cassandra:
   https://cassandra.apache.org/_/download.html
3. Start Cassandra server:

```
bin/cassandra -f
```

4. Open the CQL shell:

```
bin/cqlsh
```

Data Creation and Query

```
-- Create a keyspace
CREATE KEYSPACE company WITH replication = {
  'class': 'SimpleStrategy',
  'replication_factor': 1
};

-- Switch to the keyspace
USE company;

-- Create a table
CREATE TABLE employees (
  emp_id UUID PRIMARY KEY,
  name TEXT,
  department TEXT,
  salary DOUBLE
);

-- Insert data
INSERT INTO employees (emp_id, name, department, salary)
VALUES (uuid(), 'Bob', 'IT', 75000);

-- Query all records
SELECT * FROM employees;
```

Explanation of Commands

- `CREATE KEYSPACE`: Creates a logical namespace for tables.
- `CREATE TABLE`: Defines a table with columns and a primary key.
- `INSERT INTO`: Adds a row to the table.
- `SELECT`: Retrieves data from the table.

3. Neo4j – Graph-Based Database

Explanation

Neo4j is a graph database that stores data as nodes (entities), edges (relationships), and properties (attributes). It is ideal for highly connected data such as social networks, fraud detection, and recommendation engines.

Installation Steps

1. Download Neo4j Desktop:
   https://neo4j.com/download/
2. Install and open Neo4j Desktop.
3. Create and start a new local database.
4. Access the database via browser:
   http://localhost:7474

Data Creation and Query

```
// Create two employee nodes and one department node
CREATE (a:Employee {name: 'Alice'})
CREATE (b:Employee {name: 'Bob'})
CREATE (d:Department {name: 'Sales'})

// Create relationships between them
CREATE (a)-[:WORKS_IN]->(d)
CREATE (b)-[:COLLEAGUE_OF]->(a)

// Query to retrieve employees and departments
MATCH (e:Employee)-[:WORKS_IN]->(d:Department)
RETURN e.name, d.name
```

Explanation of Commands

- `CREATE`: Adds nodes or relationships to the graph.
- `:Employee` / `:Department`: Denote node labels (types).

- `[:WORKS_IN]`: Defines a directed relationship from one node to another.
- `MATCH` and `RETURN`: Used to retrieve patterns and return specific fields.

Result

- MongoDB successfully stored and queried JSON-style documents.
- Cassandra created a keyspace and table, and handled tabular data with wide columns.
- Neo4j created nodes and relationships, then retrieved connected data using graph patterns.