Graph database with Neo4j

Study Points Assignment on Databases SOFT 2024

Data Ingestion

We have inserted data from our four csv files (see data folder) into one graph by the following cypher queries:

```
LOAD CSV WITH HEADERS FROM 'file:///countries.csv' AS row
CREATE (:Country {name: row.Country, location: row['Country
Location']})

LOAD CSV WITH HEADERS FROM 'file:///cities.csv' AS row
MATCH (country:Country) WHERE ID(country) =
toInteger(row.Country)
CREATE (city:City {name: row.City, location: row['City
Location'], c40: row.C40 = 'True'})
CREATE (city)-[:IN_COUNTRY]->(country)

LOAD CSV WITH HEADERS FROM 'file:///emissions.csv' AS row
MATCH (city:City) WHERE ID(city) = toInteger(row.City)
CREATE (emission:Emission {year: toInteger(row['Reporting
Year']), totalEmissions: toFloat(row['Total Emissions'])})
CREATE (emission)-[:EMISSION_FOR]->(city)

LOAD CSV WITH HEADERS FROM 'file:///reductions.csv' AS row
MATCH (city:City) WHERE ID(city) = toInteger(row.City)
CREATE (reduction:Reduction {
  year: toInteger(row['Reporting Year']),
  baselineYear: toInteger(row['Baseline year']),
  baselineEmissions: toFloat(row['Baseline emissions']),
  targetReduction: toFloat(row['Percentage reduction target']),
  targetDate: toInteger(row['Target date'])
})
CREATE (reduction)-[:REDUCTION_FOR]->(city)
```

1. Operations for Maintenance of the Database

Data Backup: To automate data backups in Neo4j, you can set up a cron job on the server where Neo4j is hosted. Here's an example of a cron job that takes a daily backup at 1 AM:

```
0 1 * * * /path/to/neo4j/bin/neo4j-admin backup --backup-dir=/path/to/backup/dir
--database=neo4j --pagecache=2G
```

Performance Monitoring: Enable and configure monitoring in neo4j.conf:

```
dbms.logs.query.enabled=VERBOSE
dbms.logs.query.threshold=200ms
metrics.enabled=true
metrics.prometheus.endpoint=localhost:2004
```

Index Management: Regularly review and create indexes based on query patterns:

```
CREATE INDEX ON :City(name);
CREATE INDEX ON :Country(name);
```

Data Integrity Checks: Schedule regular integrity checks using APOC procedures:

```
0 5 * * * /path/to/neo4j/bin/neo4j-admin check-consistency --database=neo4j
--verbose=true
```

2. Formulate Questions for Data Extraction

1. What are the total emissions by year for each country?
2. Which city had the highest reduction in emissions compared to its baseline year?
3. How do emissions trends compare between C40 and non-C40 cities?

4. List all cities in Denmark with their respective total emissions for 2020.
5. What is the percentage reduction in emissions achieved by each city since the baseline year?
6. Which country has the most cities participating in emission reduction programs?
7. Detail the emission and reduction data for cities with more than 1 million population.
8. What are the forecasted reductions for the next decade under current trends?
9. Compare baseline emissions to current emissions for cities with severe pollution issues.
10. Identify relationships between economic indicators (if available) and emission levels.

Implementing Information Extraction:

Use Cypher queries to extract this data.
For instance, for question 1:

```
MATCH (c:City)-[r:EMISSION_FOR]->(e:Emission)
RETURN c.name, e.year, sum(e.totalEmissions) as TotalEmissions
GROUP BY e.year, c.name
ORDER BY e.year
```

3. Scaling the Database Model

Considerations:

ACID vs. CAP Theorem: Neo4j strongly supports ACID properties. In distributed environments, it's important to consider trade-offs between consistency, availability, and partition tolerance (CAP).

Replication and Sharding: Implement data replication for fault tolerance and sharding for horizontal scaling.

Implementation:

Set up a causal cluster for replication and failover:

neo4j.conf on all cluster members:

```
dbms.mode=CORE
causal_clustering.expected_core_cluster_size=3
causal_clustering.discovery_type=LIST
causal_clustering.initial_discovery_members=core1:5000,core2:5000,core3:5000
```

4. Validate and Test Database Operations

Unit Testing: Use the Neo4j testing library for writing unit tests for Cypher queries:

Example in java:

```
try (Driver driver = GraphDatabase.driver(uri,
AuthTokens.basic(user, password));
     Session session = driver.session()) {
    String result = session.writeTransaction(tx -> {
        Result res = tx.run("CREATE (n:Node) RETURN n");
        return res.single().get(0).asString();
    });
    assert result.contains("Node");
}
```

Integration Testing: Test the integration of Neo4j with applications.

Load Testing: Simulate concurrent access to evaluate performance under stress. Tools like Apache JMeter can simulate multiple users or queries accessing the database concurrently.

5. Evaluate Database Performance

Monitoring Tools: Use Neo4j's query log, explain plans, and profiling features to identify slow queries and bottlenecks.

Optimization: Optimize queries, adjust database configurations, and scale resources based on findings.

Analyze slow queries using PROFILE:

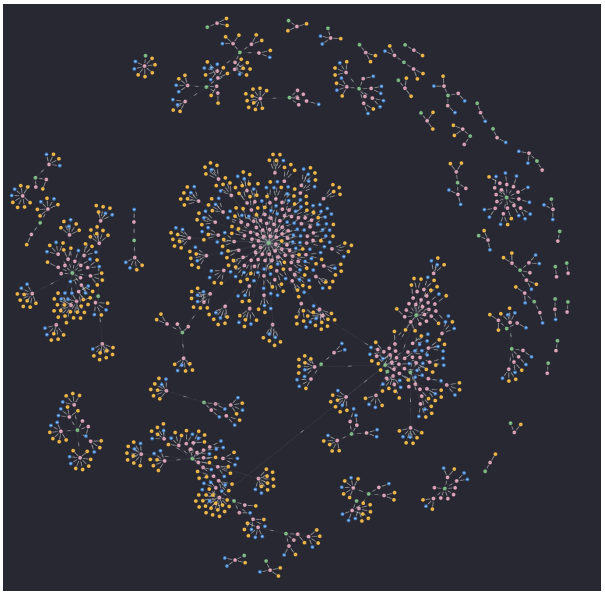
```
PROFILE MATCH (c:City)-[:EMISSION_FOR]->(e:Emission)
RETURN c.name, sum(e.totalEmissions)
GROUP BY c.name
```

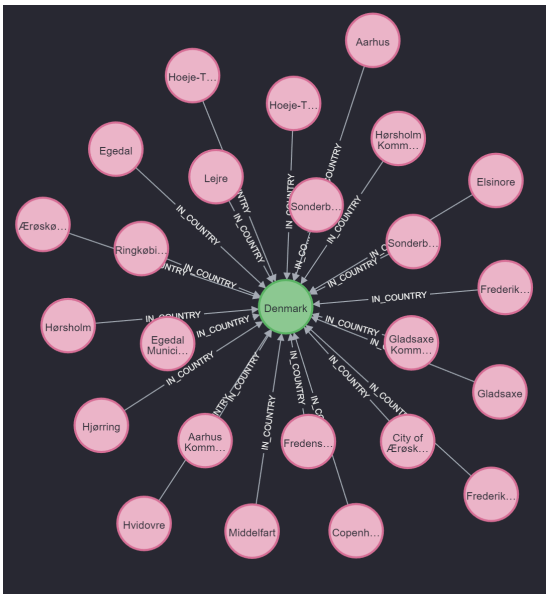
6. Documentation

Process Documentation: Use tools like Lucidchart or Microsoft Visio to create ER diagrams, architecture diagrams, and flowcharts.

Operation Documentation: Document the maintenance operations, backup procedures, and testing strategies.

Below are pictures of the graph and a subgraph with nodes related to the Country-node “Denmark”:





7. Formulate Conclusions and Recommendations

We really like the Neo4J desktop UI. It feels more straightforward to use compared to f.x MSSQL SSMS and MySql workbench.

We came up with these pros and cons:

Pros:

- Flexibility
- Graphical visual presentation
- Fast execution (connected data makes it easier to retrieve and navigate)
- No need for primary ID

Cons:

- Less Scalability (Horizontally)
- Declaring data types not available
- Performance issues with larger datasets.