Assignment No 12

Name-Sujan Mujawar

PRN-21510048

Spatial and Geographic Data Geospatial is the natural domain for Graph Database Use Neo4j and Neo4j Spatial

Problem Statement : Finding Things Close to Other Things.

Application in : location-based services on the web

-→

Spatial and Geographic Data Geospatial is the natural domain for Graph Database Use Neo4j and Neo4j Spatial

Introduction:

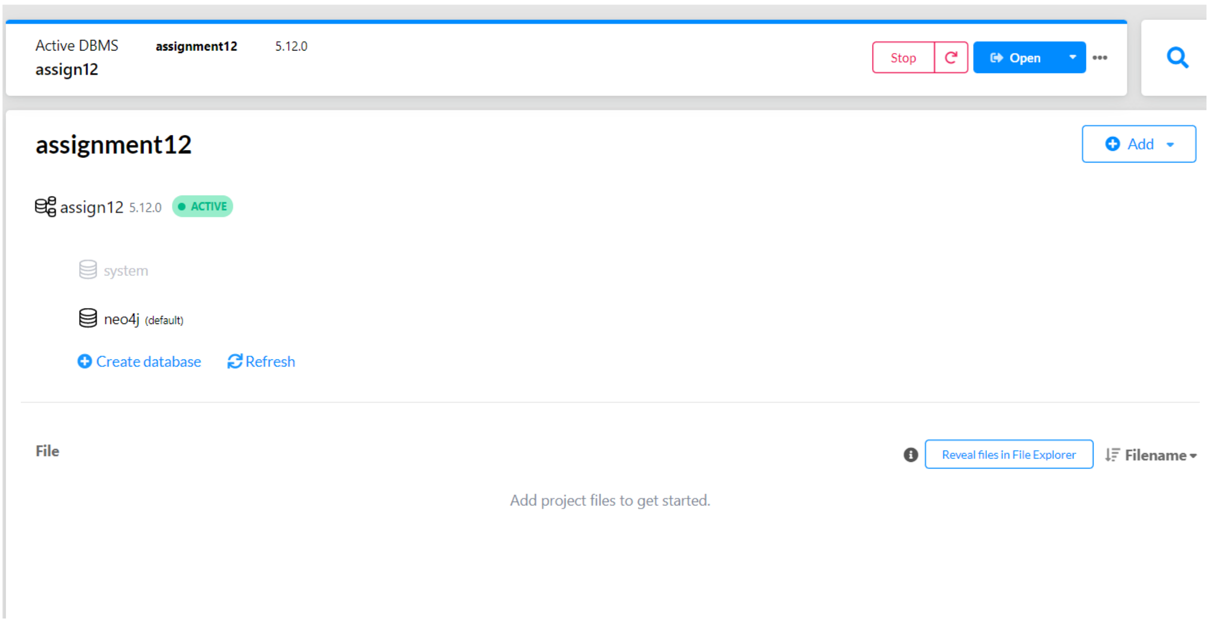
Neo4j is a native graph database platform, built from the ground up leverage not only data but also data relationships. Neo4j connects data as it's stored, enabling queries never before imagined, at speeds never thought possible. Geography is a natural domain for graphs and graph databases.So natural, in fact, that early map users of Neo4j simply rolled their own map support. However, it takes some effort to deal with spatial indexes, geometries and topologies, and so, since September 2010, the Neo4j Spatial project has been providing early access releases enabling a wide range of convenient and powerful geographic capabilities in the Neo4j database.

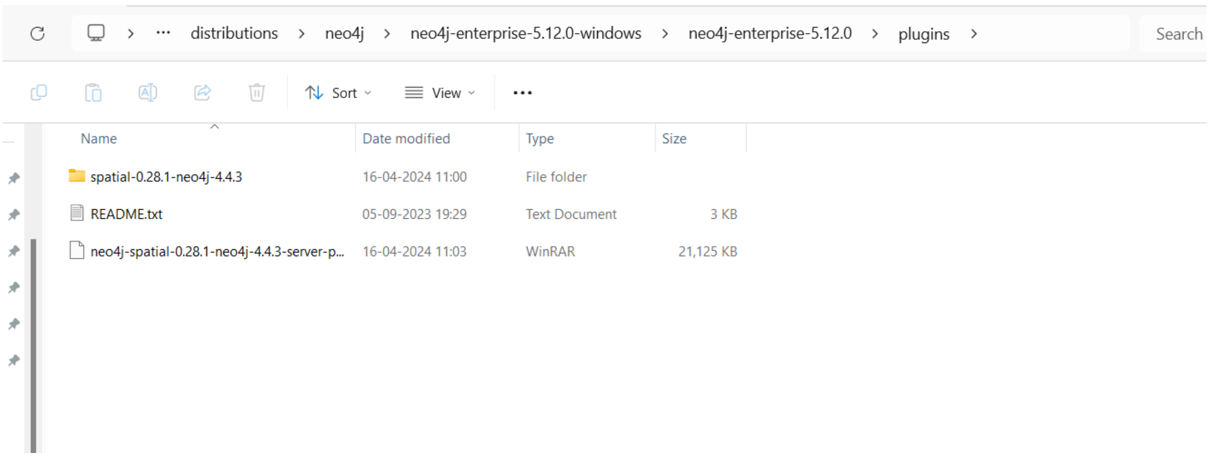
Theory:

Today's CIOs and CTOs don't just need to manage larger volumes of data they need to generate insight from their existing data. In this case, the relationships between data points matter more than the individual points themselves. In order to leverage data relationships, organizations need a database technology that stores relationship information as a first-class entity. That technology is a graph database. Ironically, legacy relational database management systems (RDBMS) are poor at handling data relationships. Their rigid schemas make it difficult to add different connections or adapt to new business requirements. Not only do graph databases effectively store data relationships; they're also flexible when expanding a data model or conforming to changing business needs. One of the simplest and most intuitive places to start is to ask the question: How do I find things close to other things? This is exactly the question answered by location-based services on the web, as well as a number of existing spatial databases. In the NoSQL area, CouchDD released GeoCouch in 2009, and MongoDD released their geohashing index in 2010. Both answer exactly this question. Unlike these other NoSQL databases, Neo4j started with support for complex geometries in 2010. While simple proximity searches have been possible, they have only recently become simple and intuitive.

Procedure: 1.

Install the Neo4jSpatial (https://github.com/neo4jcontrib/spatial) from GitHub. It is the Neo4j plug-in that facilitates geospatial operations on data stored in Neo4j. Unzip that zip/rar file and paste in plugin folder in neo4j cluster and restart neo4j server.



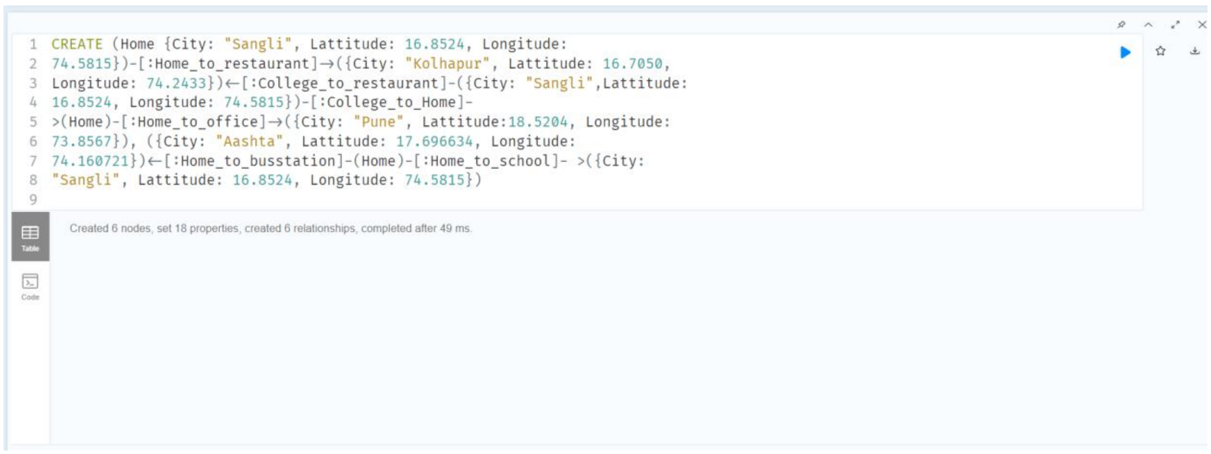


distributions › neo4j › neo4j-enterprise-5.12.0-windows › neo4j-enterprise-5.12.0 › plugins ›
Search
Sort
View

Name	Date modified	Type	Size
spatial-0.28.1-neo4j-4.4.3	16-04-2024 11:00	File folder	
README.txt	05-09-2023 19:29	Text Document	3 KB
neo4j-spatial-0.28.1-neo4j-4.4.3-server-p...	16-04-2024 11:03	WinRAR	21,125 KB

Run this Cypher query in neo4j database

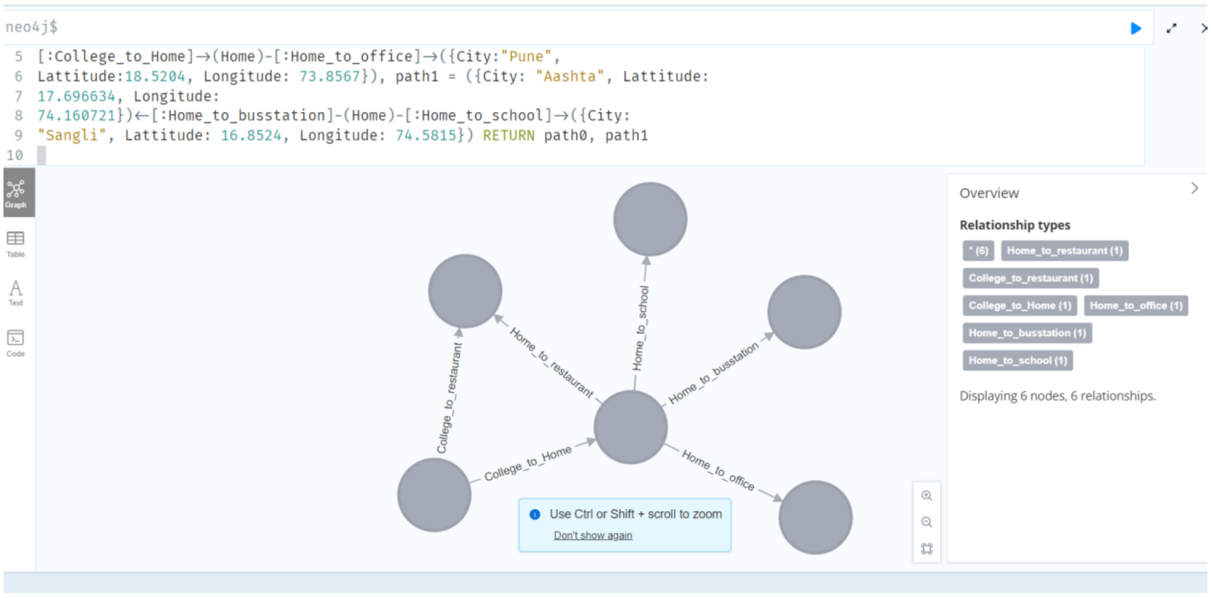
1. CREATE (Home {City: "Sangli", Lattitude: 16.8524, Longitude: 74.5815})-[:Home_to_restaurant]->({City: "Kolhapur", Lattitude: 16.7050, Longitude: 74.2433})<-[:College_to_restaurant]-({City: "Sangli",Lattitude: 16.8524, Longitude: 74.5815})-[:College_to_Home]- >(Home)-[:Home_to_office]->({City: "Pune", Lattitude:18.5204, Longitude: 73.8567}), ({City: "Aashta", Lattitude: 17.696634, Longitude: 74.160721})<-[:Home_to_busstation]-(Home)-[:Home_to_school]- >({City: "Sangli", Lattitude: 16.8524, Longitude: 74.5815})



```
CREATE (Home {City: "Sangli", Lattitude: 16.8524, Longitude:
74.5815})-[:Home_to_restaurant]→({City: "Kolhapur", Lattitude: 16.7050,
Longitude: 74.2433})←[:College_to_restaurant]-({City: "Sangli",Lattitude:
16.8524, Longitude: 74.5815})-[:College_to_Home]-
>(Home)-[:Home_to_office]→({City: "Pune", Lattitude:18.5204, Longitude:
73.8567}), ({City: "Aashta", Lattitude: 17.696634, Longitude:
74.160721})←[:Home_to_busstation]-(Home)-[:Home_to_school]- >({City:
"Sangli", Lattitude: 16.8524, Longitude: 74.5815})
```

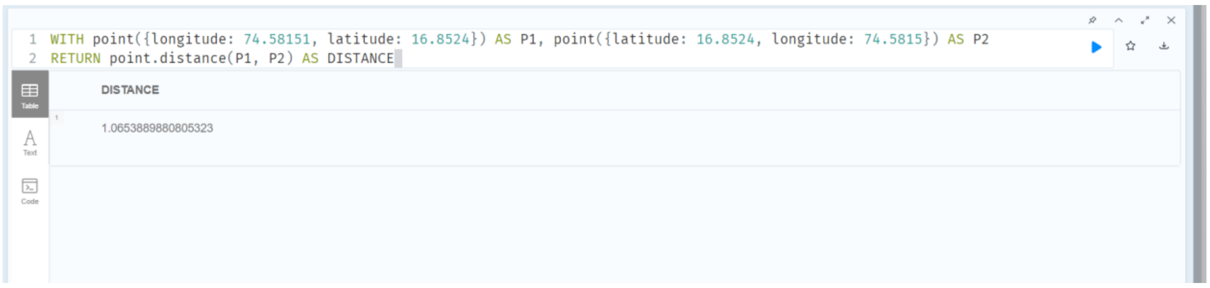
```
Created 6 nodes, set 18 properties, created 6 relationships, completed after 49 ms.
```

2. MATCH path0 = (Home {City: "Sangli", Lattitude: 16.8524, Longitude: 74.5815})-[:Home_to_restaurant]->({City: "Kolhapur", Lattitude: 16.7050, Longitude: 74.2433})<-[:College_to_restaurant]-({City: "Sangli",Lattitude: 16.8524, Longitude: 74.5815})-[:College_to_Home]->(Home)-[:Home_to_office]->({City:"Pune", Lattitude:18.5204, Longitude: 73.8567}), path1 = ({City: "Aashta", Lattitude: 17.696634, Longitude: 74.160721})<-[:Home_to_busstation]-(Home)-[:Home_to_school]->({City: "Sangli", Lattitude: 16.8524, Longitude: 74.5815}) RETURN path0, path1



```
neo4j$
[:College_to_Home]→(Home)-[:Home_to_office]→({City:"Pune",
Lattitude:18.5204, Longitude: 73.8567}), path1 = ({City: "Aashta", Lattitude:
17.696634, Longitude:
74.160721})←[:Home_to_busstation]-(Home)-[:Home_to_school]→({City:
"Sangli", Lattitude: 16.8524, Longitude: 74.5815}) RETURN path0, path1
```

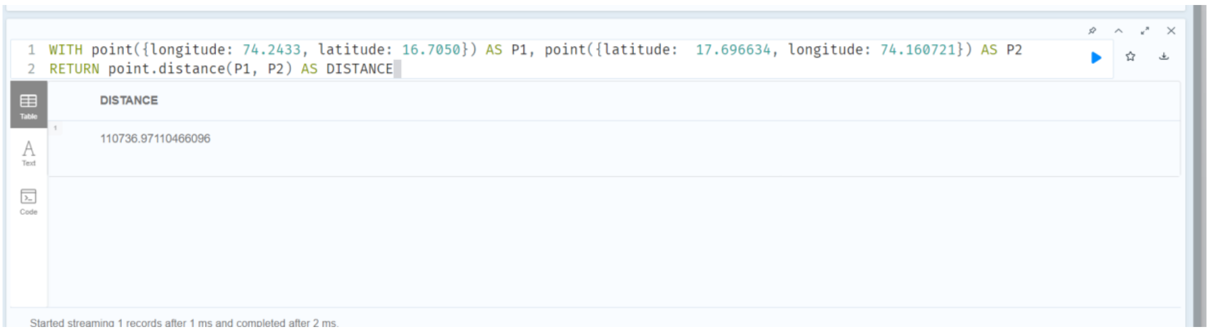
1. With point() and distance() functions finding the distance between locations by taking home as reference point:



```
WITH point({longitude: 74.58151, latitude: 16.8524}) AS P1, point({latitude: 16.8524, longitude: 74.5815}) AS P2
RETURN point.distance(P1, P2) AS DISTANCE
```

```
DISTANCE
1.0653889880805323
```

Distance between restaurant in Kolhapur and bus station in Aasht



```
WITH point({longitude: 74.2433, latitude: 16.7050}) AS P1, point({latitude:  17.696634, longitude: 74.160721}) AS P2
RETURN point.distance(P1, P2) AS DISTANCE
```

```
DISTANCE
110736.97110466096
```

```
Started streaming 1 records after 1 ms and completed after 2 ms.
```

Results: With distance() and point() functions we found out the distance between restaurant in chale and bus station in Pandharpur. And found many other distances from home to other places.

Conclusion: With the Neo4J graph database and geospatial data, we find the close/ nearest place with respect to reference points in an efficient way