Exploring NoSQL Databases: Challenges and Opportunities

CS 5614 Spring 2024

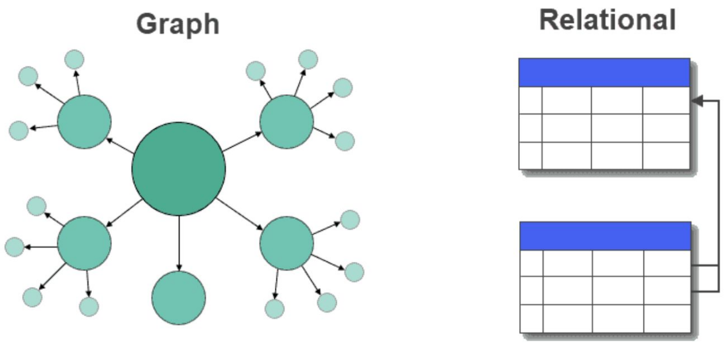
Sindhuja Madabushi

NoSql Databases

- NoSQL databases have prominence in the era of Generative AI.
- They are designed to handle large volumes of unstructured or semi-structured data.
- Focus on how NoSQL databases address the unique needs of handling graph data.

NoSql Databases (Cont.)

- Document and Key-Value models - quick data retrieval.
- Graph models - complex relationships.
- *Choose the data model based on the specific requirements of your application.*



https://phoenixnap.com/kb/graph-database

There are many DBs out there

CouchDB
relax

Redis

Neo4j
the graph database

Cassandra

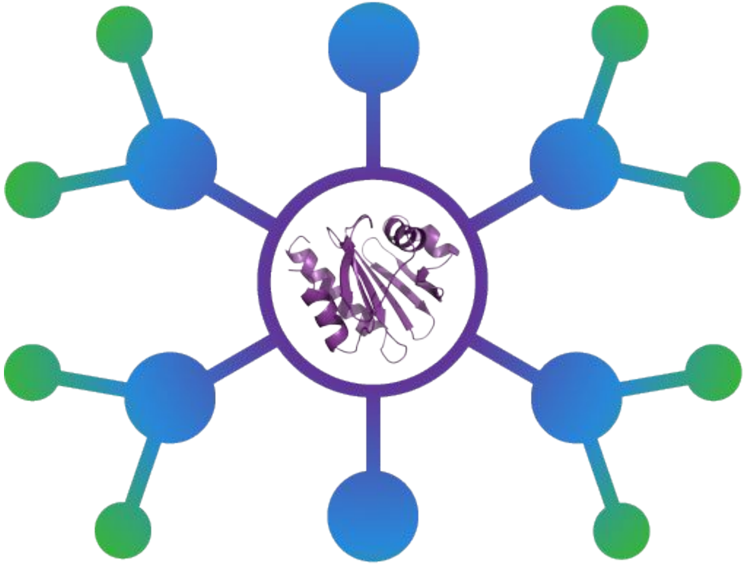
mongoDB

membase

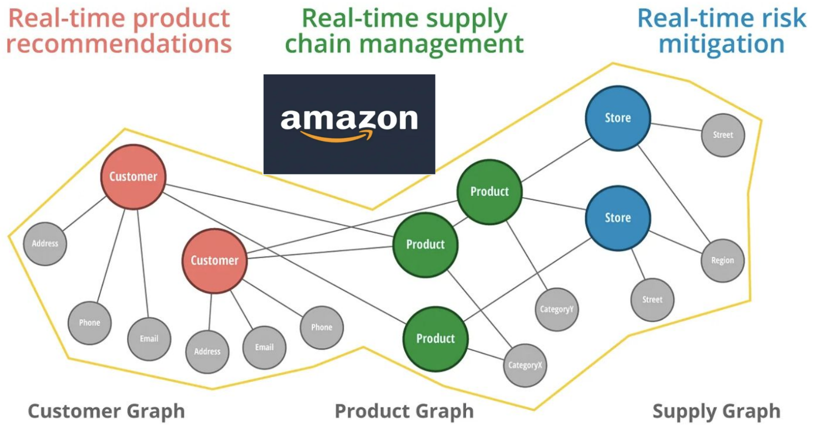
riak

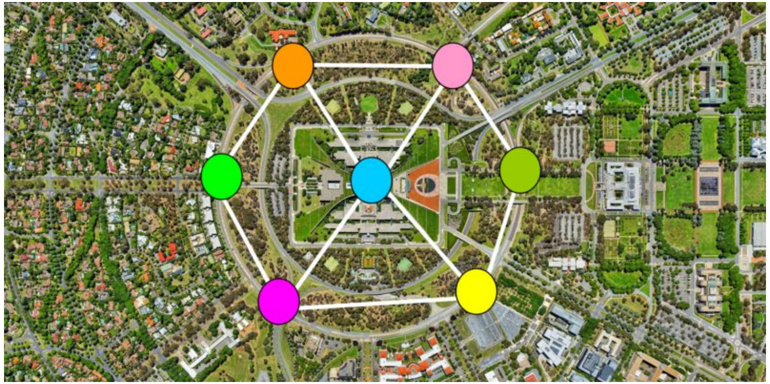
Graph DBs are designed to store and query graph data

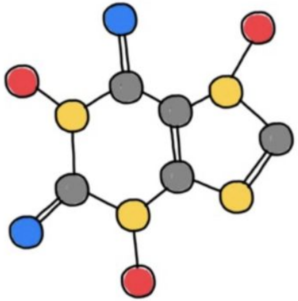
- Nodes: Represent entities or objects in the data.
- Edges: Represent the relationships between nodes.

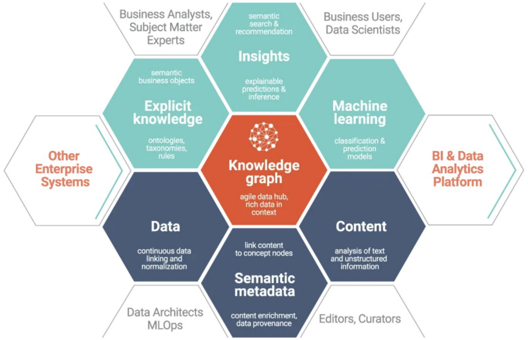


Credit: https://zhanggroup.org/PEPPI/

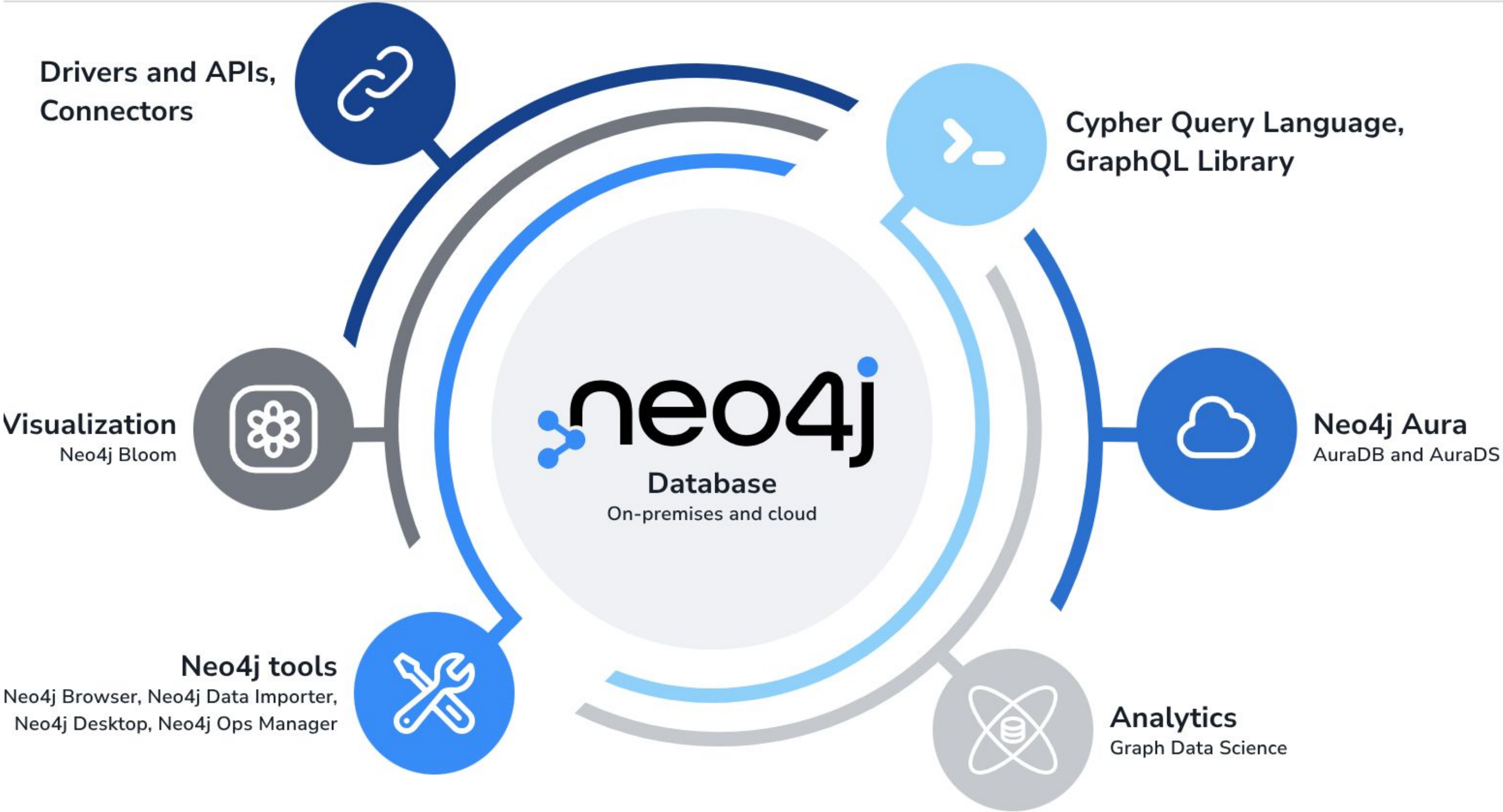








https://www.freecodecamp.org/news/deep-dive-into-graph-traversals-227a90c6a261/
https://rajshah001.medium.com/graphs-and-real-life-application-28759b77b833
https://www.freecodecamp.org/news/data-structures-101-graphs-a-visual-introduction-for-beginners-6d88f36ec768/
https://medium.com/analytics-vidhya/social-network-analytics-f082f4e21b16
https://rajshah001.medium.com/graphs-and-real-life-application-28759b77b833



https://neo4j.com/docs/getting-started/

Graph Data Sources

- LDBC Datagen
- SNAP

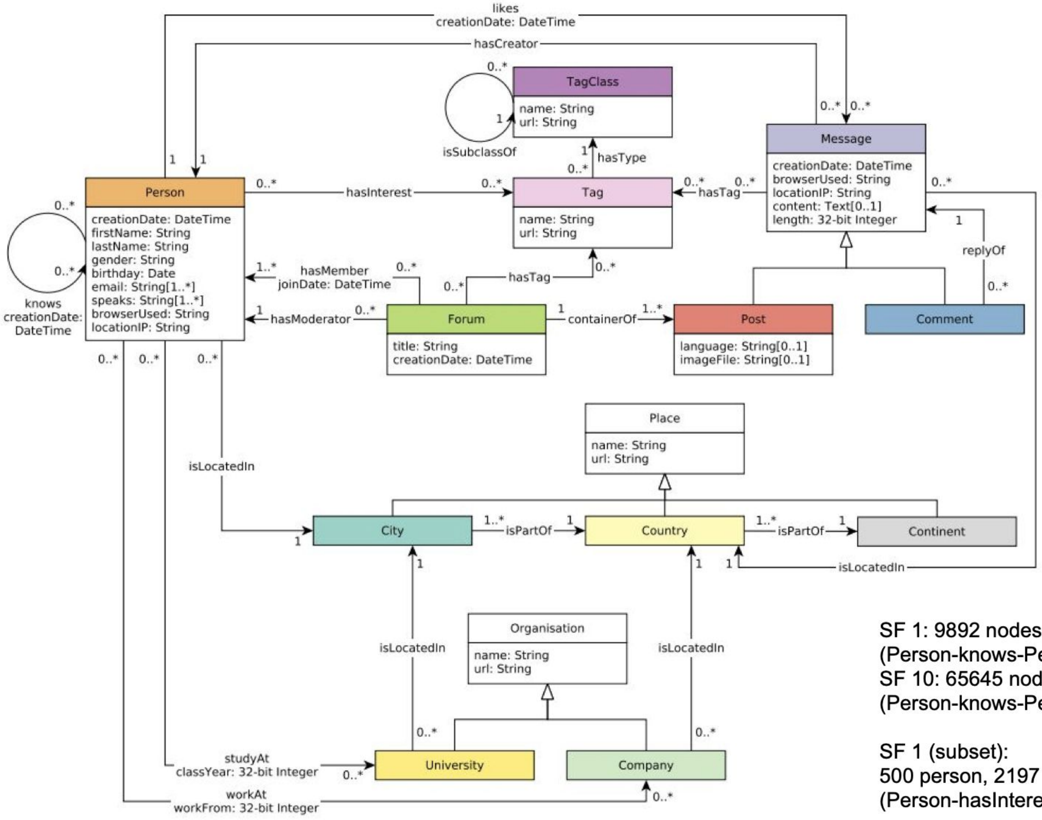
Stanford Large Network Dataset Collection

- Social networks : online social networks, edges represent interactions between people
- Networks with ground-truth communities : ground-truth network communities in social and information networks
- Communication networks : email communication networks with edges representing communication
- Citation networks : nodes represent papers, edges represent citations
- Collaboration networks : nodes represent scientists, edges represent collaborations (co-authoring a paper)
- Web graphs : nodes represent webpages and edges are hyperlinks
- Amazon networks : nodes represent products and edges link commonly co-purchased products
- Internet networks : nodes represent computers and edges communication
- Road networks : nodes represent intersections and edges roads connecting the intersections
- Autonomous systems : graphs of the internet
- Signed networks : networks with positive and negative edges (friend/foe, trust/distrust)
- Location-based online social networks : social networks with geographic check-ins
- Wikipedia networks, articles, and metadata : talk, editing, voting, and article data from Wikipedia
- Temporal networks : networks where edges have timestamps
- Twitter and Memetracker : memetracker phrases, links and 467 million Tweets
- Online communities : data from online communities such as Reddit and Flickr
- Online reviews : data from online review systems such as BeerAdvocate and Amazon
- User actions : actions of users on social platforms.
- Face-to-face communication networks : networks of face-to-face (non-online) interactions
- Graph classification datasets : disjoint graphs from different classes
- Computer communication networks : communications among computers running distributed applications
- Cryptocurrency transactions : transactions covering several cryptocurrencies and exchanges
- Telecom networks : relationships between users, packages, apps, and cells in a telecom network

SNAP networks are also available from SuiteSparse Matrix Collection by Tim Davis.

https://ldbcouncil.org/post/datagen-data-generation-for-the-social-network-benchmark/
https://snap.stanford.edu/data/index.html

Dataset - LDBC SNB data schema



SF 1: 9892 nodes, 180623 edges
(Person-knows-Person)
SF 10: 65645 nodes, 1947294 edges
(Person-knows-Person)

SF 1 (subset):
500 person, 2197 tags, 10157 edges
(Person-hasInterest-Tag)

Cypher Query Language

Create nodes:

```
CREATE (p:Person {name: 'Alice', age: 30})
CREATE (p:Person {name: 'Bob', age: 35})
```

Create relationship between the nodes:

```
MATCH (a:Person {name: 'Alice'}), (b:Person {name: 'Bob'})
CREATE (a)-[:KNOWS]->(b)
```

Select all pairs of people who know each other

```
MATCH (p1:Person)-[r:KNOWS]->(p2:Person) RETURN p1, r, p2
```

This creates a KNOWS relationship with a property since indicating the year since Alice knows Bob:

```
MATCH (a:Person {name: 'Alice'}), (b:Person {name: 'Bob'})
CREATE (a)-[r:KNOWS {since: 2021}]->(b) RETURN r
```

Cypher Query Language - Queries

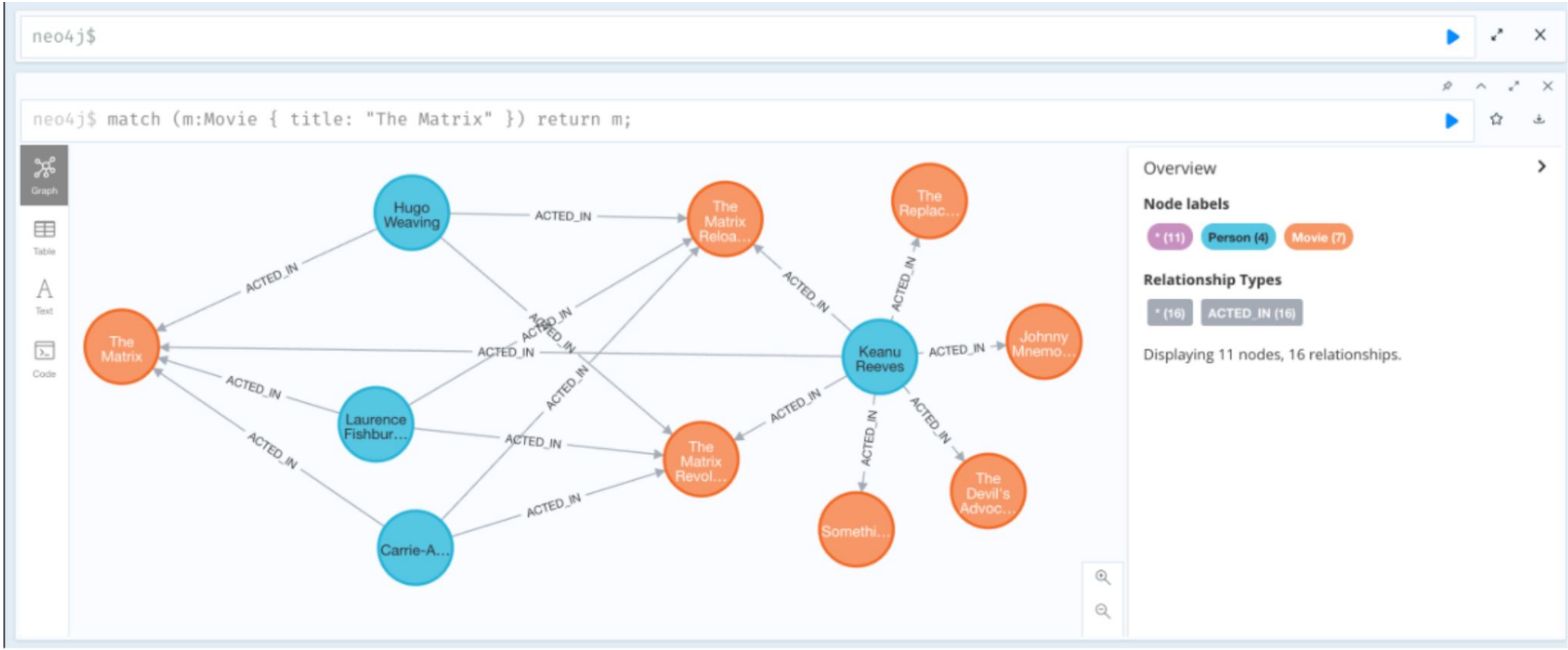
betweenness centrality:

```
CALL algo.betweenness.stream('Person','KNOWS',direction:'out')
YIELD nodeId, centrality
MATCH (user:Person) WHERE id(user) = nodeId
RETURN user.id AS user,centrality
ORDER BY centrality DESC;
```

Community detection:

```
CALL algo.louvain.stream('Person', 'KNOWS', )
YIELD nodeId, community
RETURN algo.getNodeById(nodeId).id AS user, community
ORDER BY community;
```

Working With Neo4j



Cypher Query Language - Stored Procedures

Stored procedure call: CALL algo.procedure.cosine()

```
public class FullTextIndex
{
private static final Map<String,String> FULL_TEXT =
stringMap( IndexManager.PROVIDER, "lucene", "type",
"fulltext" );
@Context
public GraphDatabaseService db;

@Context
public Log log;
@Procedure(value = "similarity.procedure")
@Description("Execute lucene query in the given index,
return found
nodes")
```

```
public Stream<SearchHit> search()
{
Stream<SearchHit> s1 = null, s2;
Boolean s1Empty= true;
String queryString="";
List<String> a= new ArrayList<>()

String[] emb = {
        "0.0797428,0.182545,0.0576887,0.0351693",
        "-0.0777048,0.386052,0.584654,3.87082",
}
```

Cypher Query Language - Stored Procedures

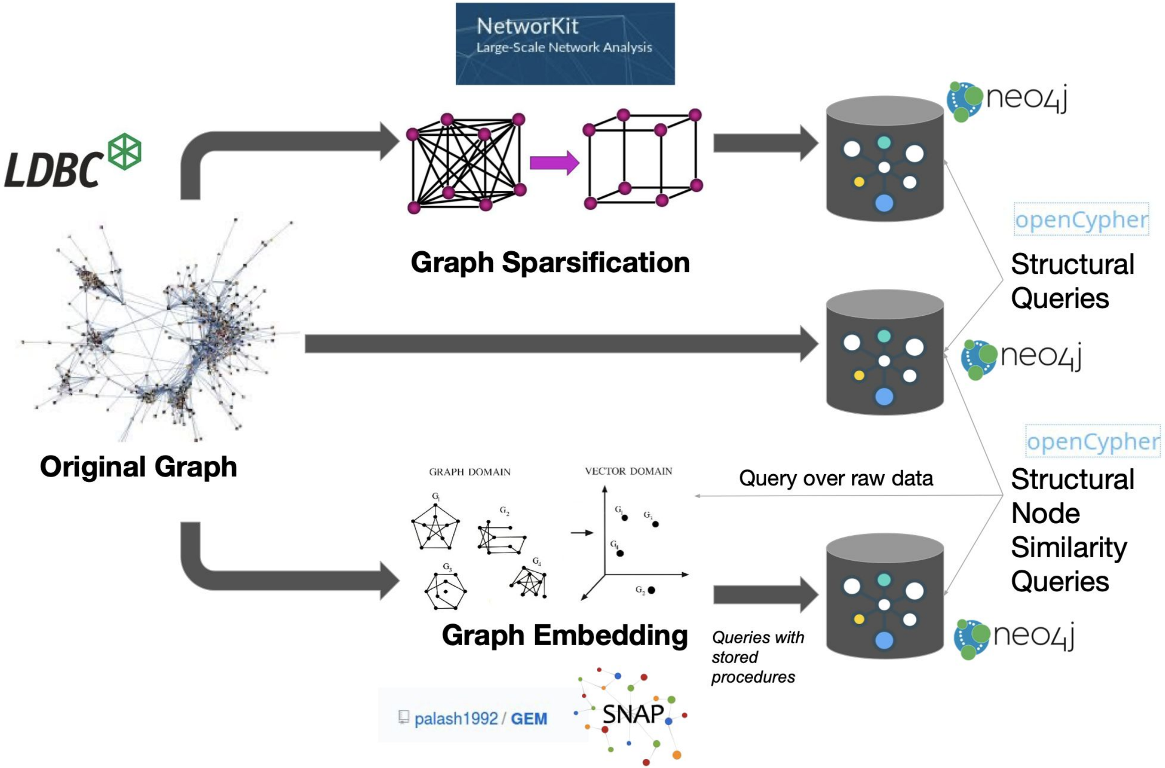
```
queryString="WITH [";
for(int i=0;i<emb.length-1;i++){
queryString+="{item: "+i+", weights: ["+emb[i]+"]}, ";
}
queryString+="{item: "+(emb.length-1)+", weights:
["+emb[emb.length-1]+"]}] as data CALL
algo.similarity.cosine.stream(data) YIELD item1, item2,
similarity RETURN item1, item2, similarity;";
s1=db.execute(queryString).stream().map(it->new
SearchHit(it.values().stream().map(it2->it2.toString()).collect(C
ollectors.joining(";"))));
return s1;
}
```

```
public static class SearchHit
{
// This records contain a single field named 'nodeId'
public String similarity;
public SearchHit( String similarity )
{
this.similarity = similarity;
}
```

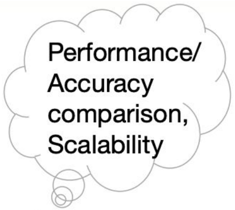
Graph Processing Benchmarks

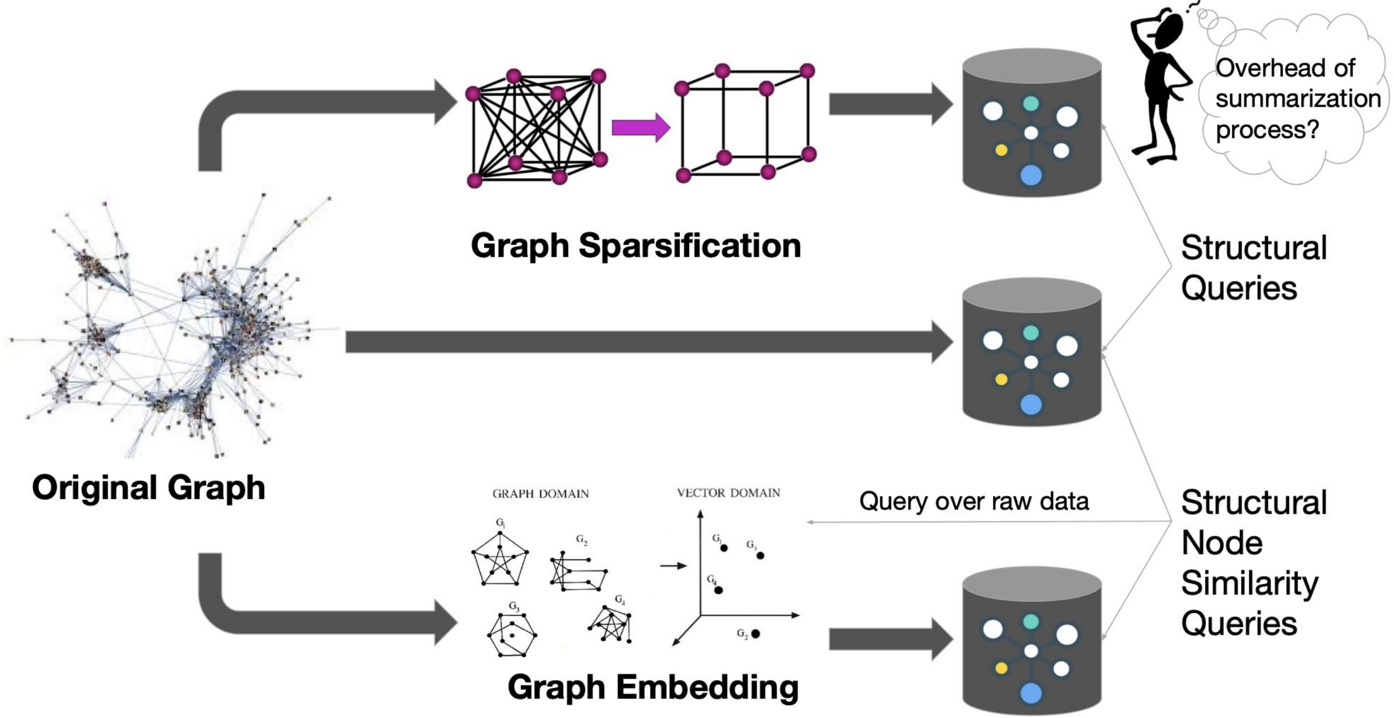
https://www.timlrx.com/blog/benchmark-of-popular-graph-network-packages-v2

Why Graph processing Techniques?

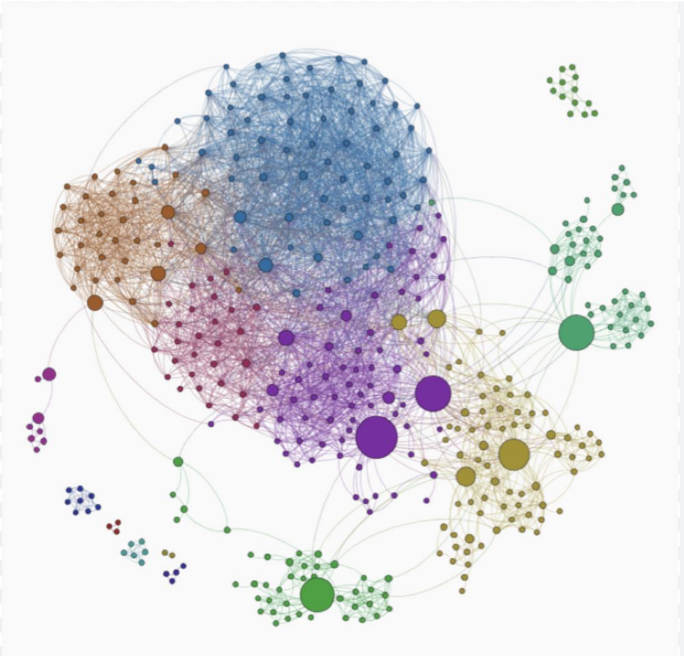


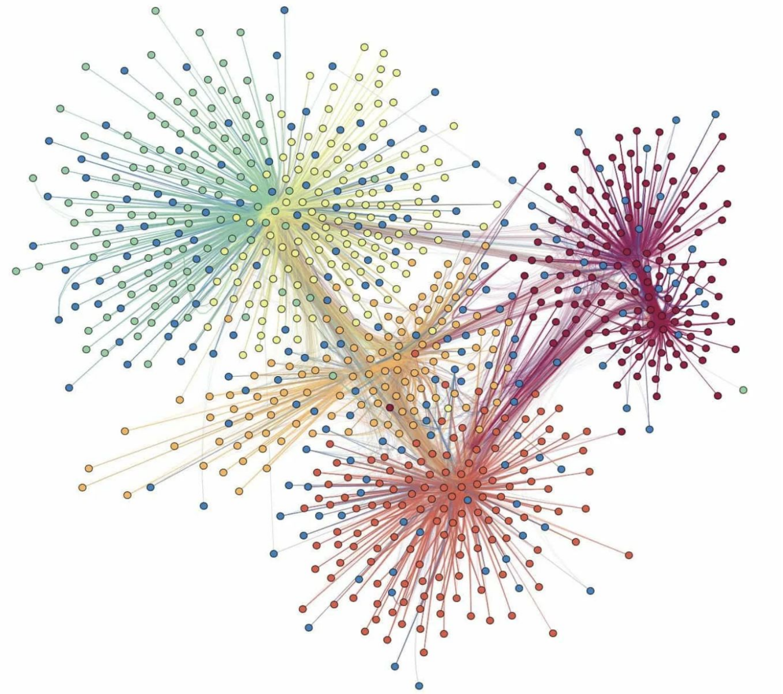
How much do these techniques improve graph processing?





Real World Graphs





https://www.pulsarplatform.com/blog/2014/detecting-communities-using-social-network-analysis/
https://towardsdatascience.com/influential-communities-in-social-network-simplified-fe5050dbe5a4

Other Graph DBs

There are 60+ graph databases:

- Amazon Neptune
- Neo4j
- OrientDB
- ArangoDB
- Elastic Search
- TitanDB

Processing capabilities - Neo4j

Cypher Query Language: Highly expressive and efficient for graph queries.

Graph Algorithms: Supports complex operations like pathfinding, centrality, and community detection.

Real-Time Processing: Enables quick data retrieval and updates for dynamic graph structures.

Indexing and Caching: Enhances performance for read-heavy workloads.

Data Import and Integration: Efficiently handles data from various sources and formats.

Can be integrated with big data with frameworks like Spark and Hadoop

Future directions

GNNs/GraphML + Generative AI

Graph DBs and Generative AI

Latest News

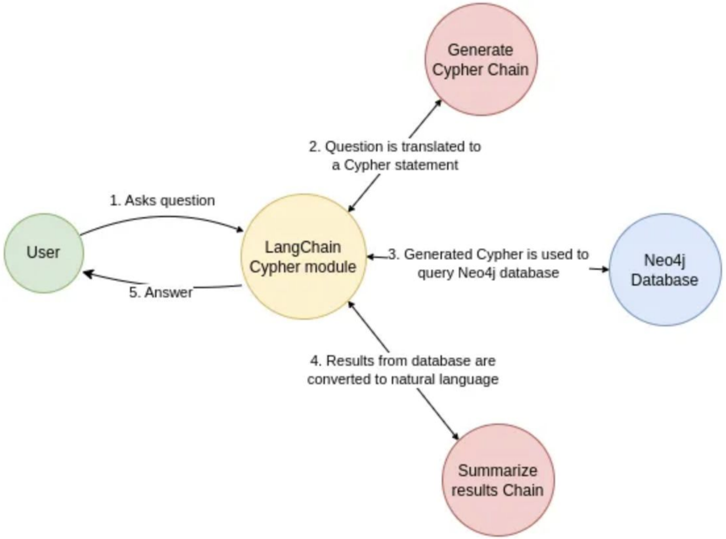
LangChain has added Cypher Search

With the LangChain library, you can conveniently generate Cypher queries, enabling an efficient retrieval of information from Neo4j.

Tomaz Bratanic · Follow

Published in Towards Data Science · 8 min read · May 24

243 9



https://towardsdatascience.com/langchain-has-added-cypher-search-cb9d821120d5

Thank You.

Questions?