✨ Member-only story

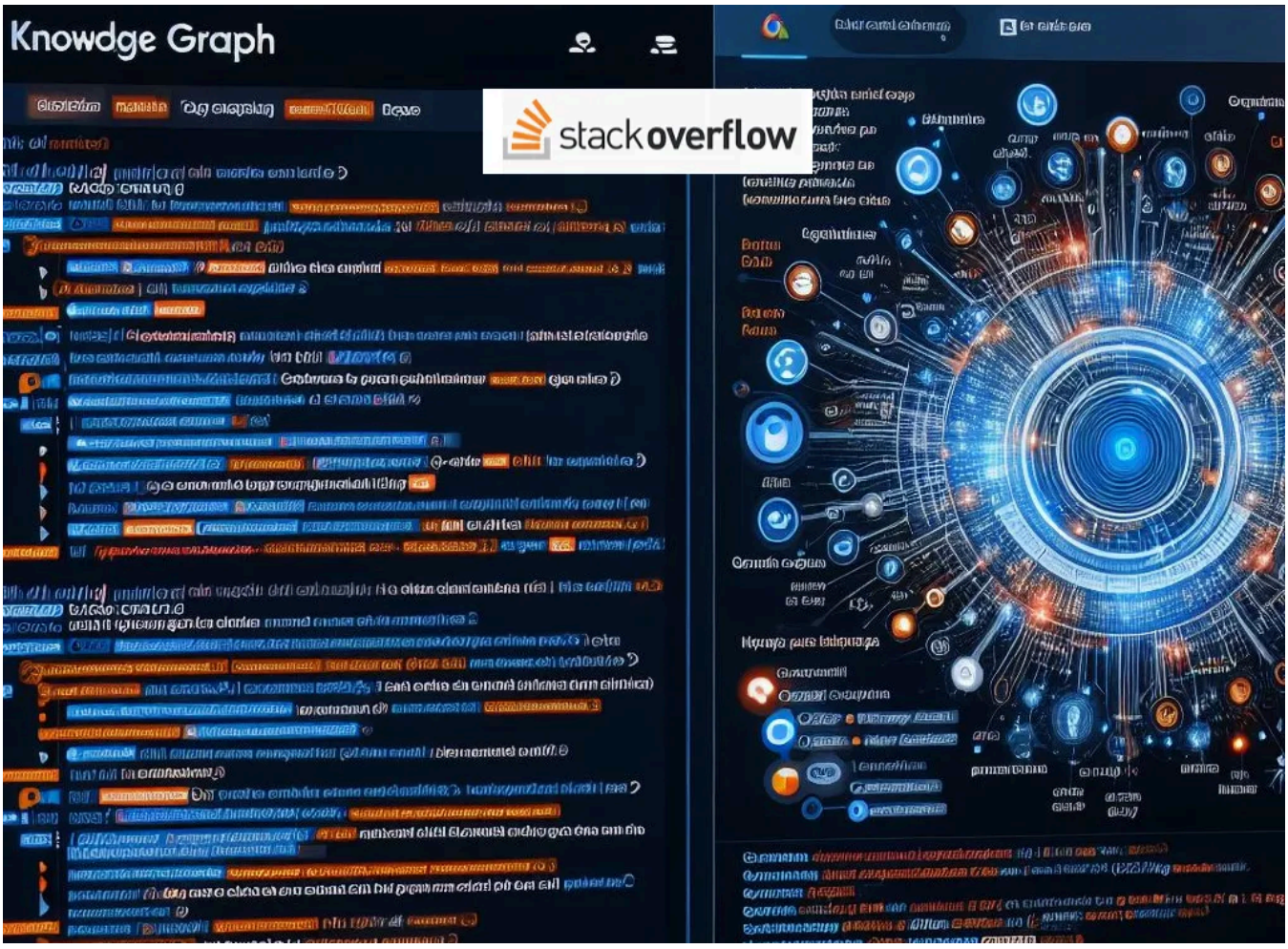
LLM Apps : Why Knowledge Graphs are super critical to know if you care about RAG : Let’s capture Wisdom of Stack Overflow in a Graph: 3

Aniket Hingane | Day Manager, Night Coder · Follow

9 min read · 3 days ago

Listen Share

Time to explore stackoverflow in different way ! KG way !



LLM Apps : Why Knowledge Graphs are super critical to know if you care about RAG : KG basics : 2

Before we build lets learn KG basics Today ! Tomorrow let’s build it !

medium.com

GitHub - aniket-work/work_w_knowledge_graph_in_spring: work_w_knowledge_graph_in_spring

work_w_knowledge_graph_in_spring. Contribute to aniket-work/work_w_knowledge_graph_in_spring development by creating an...

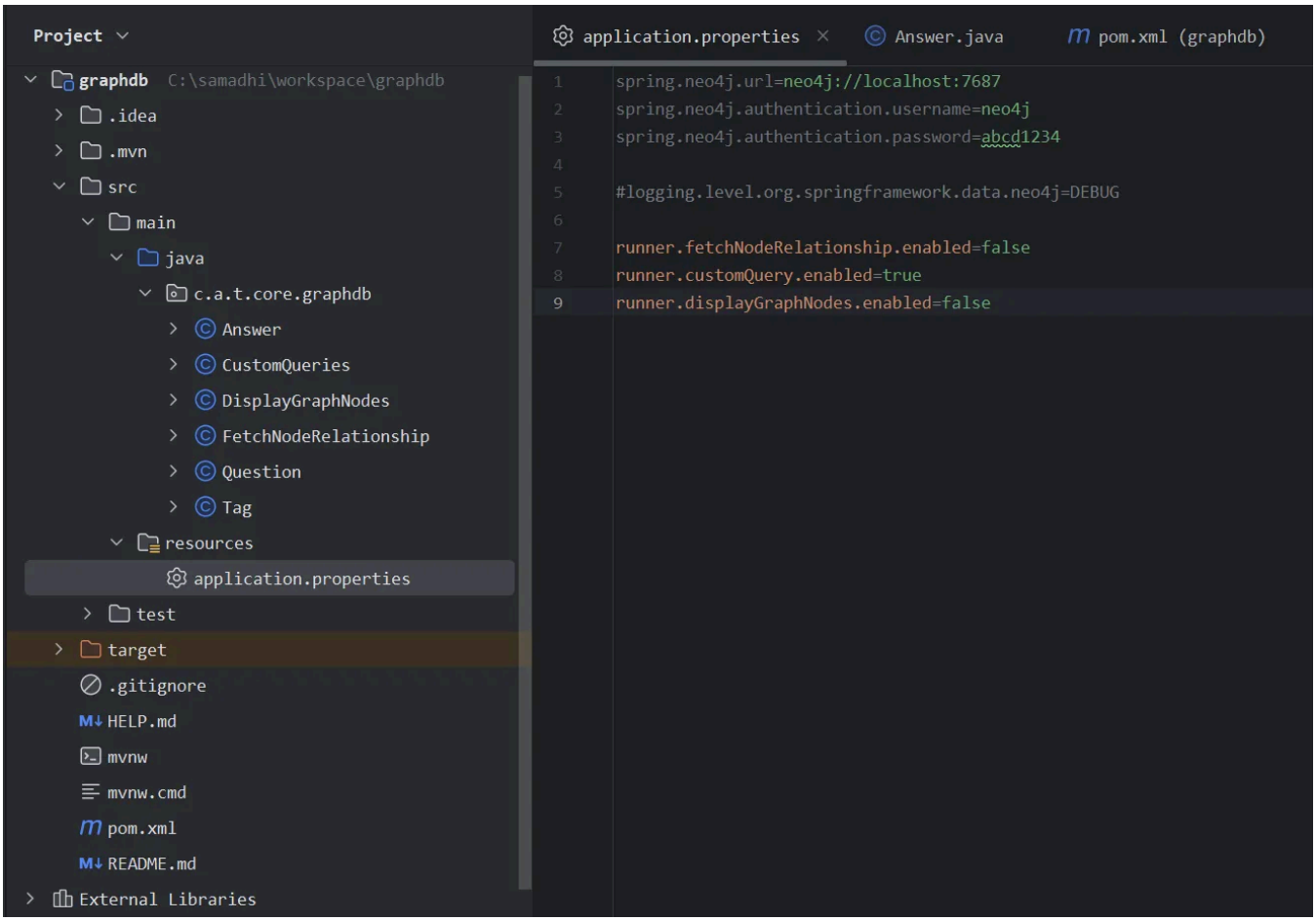
github.com

In Part 2 of our series, we delved into the basics of graph databases and how they serve as a powerful tool for managing complex relationships between data points.

Now, in Part 3, we’re taking it a step further by exploring how we can harness the power of Neo4j, a leading graph database, to navigate and analyze Stack Overflow’s vast repository of knowledge.

We’ll be leveraging the Spring Boot framework in Java to build a robust application that interacts with Neo4j, enabling us to traverse Stack Overflow’s data in a more intuitive and insightful manner.

Project Setup



Prerequisites

- Basic understanding of Java and Spring Boot (we will use java 17)
- A Neo4j database instance running (locally or remotely)

Neo4j Deployment Center

AuraDB solutions offer flexible plans for your fully managed Neo4j graph database service. Whether you are new to...

neo4j.com

- A basic dataset of Stack Overflow questions, answers, and tags (you can find ways to scrape this data online or use a smaller sample dataset to start with), we will use neo4j built in stackoverflow dataset , we will see that in few

Step 1: Project Setup

1. **Create a Spring Boot Project:** Use Spring Initializr (https://start.spring.io/) or your IDE to set up a new Spring Boot project with these dependencies:

- `spring-boot-starter-web`

- `spring-boot-starter-data-neo4j`
- `neo4j-ogm-bolt-driver`

2. Configuration: In your `application.properties` (or `.yml`) file, add details to connect to your Neo4j database: all below details, will be based on what you put in, when creating neo4j DBMS

```
spring.neo4j.url=neo4j://localhost:7687
spring.neo4j.authentication.username=neo4j
spring.neo4j.authentication.password=abcd1234
```

3. **Modeling the Knowledge Graph**

Entity classes

- **Question.java:** Represents a Stack Overflow question with properties for title, answer count, and tags.
- **Answer.java:** Represents an answer associated with a question.
- **Tag.java:** Represents a Stack Overflow tag.

Key Points:

- `@Node` annotations mark these as graph nodes.
- `@Id` and `@GeneratedValue` denote unique identifiers.
- `@Relationship` in `Question.java` establishes the "TAGGED" relationship between questions and tags.

4. **Create Spring Data Neo4j Repositories**

Interfaces: Define repository interfaces to simplify interactions with Neo4j:

```
public interface QuestionRepository extends Neo4jRepository {}

public interface TagRepository extends Neo4jRepository {}
```

```
public interface AnswerRepository extends Neo4jRepository {}
```

5. **Custom Queries (Optional):** If needed, add custom query methods using the `@Query` annotation, like in `CustomQueries` class.

6. **Designing the REST API (TODO: future work)**

Controller: Create a `StackOverflowController` :

```
@RestController
@RequestMapping("/api/stackoverflow")
public class StackOverflowController {
    @Autowired QuestionRepository questionRepository;
    @Autowired TagRepository tagRepository;
    // ... other repositories if needed

    // API endpoints will go here...
}jaj
```

Endpoints: Implement endpoints like these:

```
// Find question by title
@GetMapping("/questions/{title}")
public Question findQuestionByTitle(@PathVariable String title) {
    return questionRepository.findById(title).get();
}

// Find questions with at least a specified number of answers
@GetMapping("/questions")
public List findQuestionsByAnswerCount(@RequestParam int minAnswers) {
    return questionRepository.query("");
}

// Get tags related to a given tag
@GetMapping("/tags/{name}/related")
 public List findRelatedTags(@PathVariable String name) {
     // ... Cypher query using similarity algorithms
 }
```

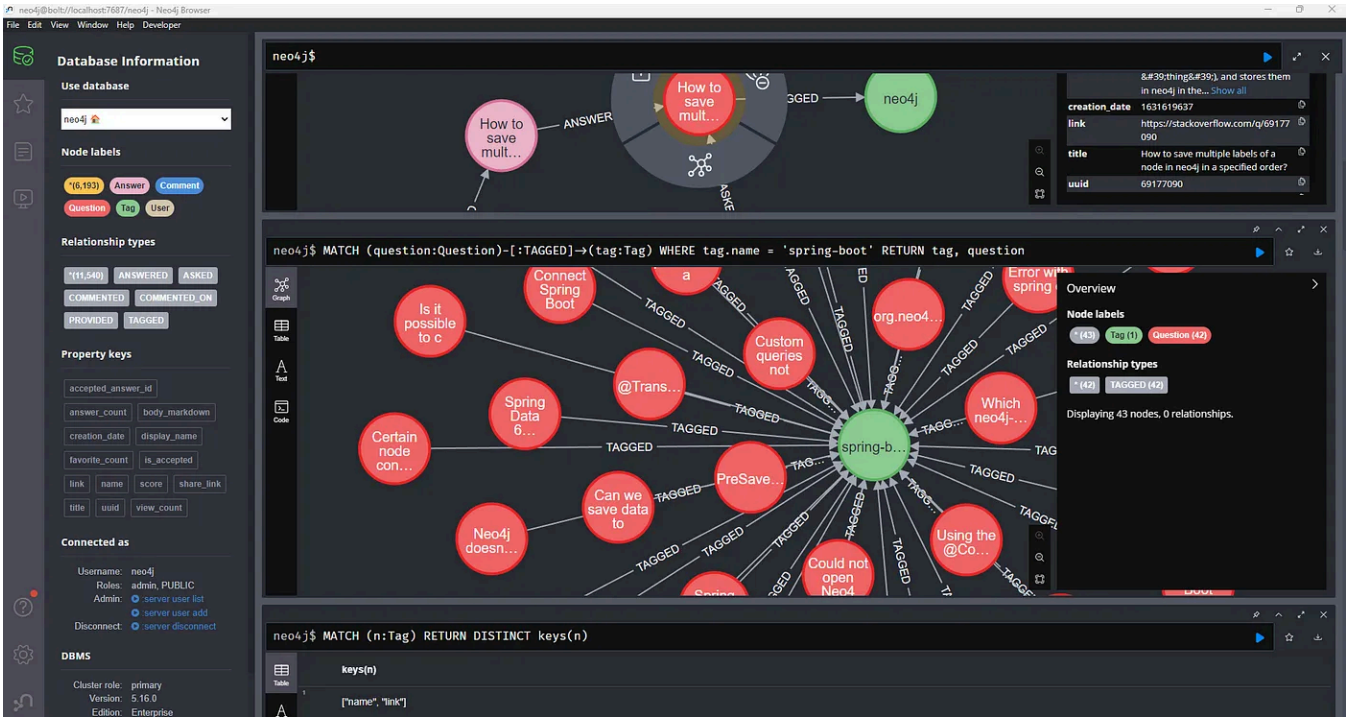
Our knowledge graph dataset

GitHub - neo4j-graph-examples/stackoverflow: Stack Overflow Questions, Answers, Tags, and Comments

Stack Overflow Questions, Answers, Tags, and Comments - GitHub - neo4j-graph-examples/stackoverflow: Stack Overflow...

github.com

> *Note : Below is my interpretation of this dataset*



Purpose :

This dataset provides a snapshot of Stack Overflow data specifically structured to showcase the power and flexibility of graph databases like Neo4j.

Key Elements :

Nodes

- **Questions:** Represents individual questions asked on Stack Overflow, likely with properties like title, body, and creation date.
- **Answers:** Represents answers provided to questions, potentially including text, scores, and associated user information.
- **Tags:** Represents the topics or categories assigned to questions.

- **Users:** Represents Stack Overflow users who have participated in asking or answering questions.

Relationships

- **ASKED:** Connects a User node to a Question they've asked.
- **ANSWERED:** Connects a User node to a Question they've answered.
- **TAGGED_WITH:** Connects Question nodes to their associated Tag nodes.

Focus :
The dataset is designed to facilitate analysis of:

- **Community Dynamics:** Understanding the interactions between users, their expertise (based on tags), and question-answering patterns.
- **Knowledge Structure:** Exploring the relationships between tags and topics to reveal hierarchies and relatedness within Stack Overflow content.
- **Trends:** Potentially identifying trends in question popularity or tag usage over time (if the dataset includes timestamps).

Important Note: The specific properties included, size of the dataset, and any additional relationships modeled might vary depending on the exact version of the dump file you use.

Let's get Cooking !

Imagine the vast knowledge of Stack Overflow — countless questions, insightful answers, and a web of interconnected topics. Our mission is to harness this knowledge and structure it as a graph database, unlocking insights that would be difficult to find in a traditional list-based format.

Mapping Questions, Answers, and Tags

Let's look at the code that makes this possible:

```
@Node("Question")
public class Question {

    @Id
    public final String title;
```

```
    @Property("answer_count")
    public final Integer quesWAnswerCount;

    @Relationship("TAGGED")
    public final List tags;

    // ... Constructor and toString() ...
}
```

The Heart of the Graph: This `Question` class is not just a code snippet; it's the blueprint for a core element in our graph. Each `Question` object will become a node in the database.

- **Properties: The Essential Details:**
- `title` : The question's unique title serves as the identifier (`@Id`).
- `quesWAnswerCount` : This stores the number of answers, giving us an immediate idea of a question's popularity.
- **Weaving Connections:** The `tags` list, marked with `@Relationship("TAGGED")` , is crucial. This is how we represent the topics associated with each question, allowing us to navigate the knowledge by tags.

The Other Key Players

Similarly, we have representations for answers (`Answer.java`) to connect them with their questions, and tags (`Tags.java`) to categorize the knowledge :

```
@Node("Answer")
public class Answer {
    @Id
    @GeneratedValue
    Long id;
    // ... other properties for the answer ...
}

@Node("Tag")
public class Tag {
    @Id
    public final String name;
```

```
        // ... Constructor and toString() ...
    }
```

Bringing it to Life with Neo4j and Spring

This code, together with the power of Neo4j (a graph database) and the convenience of Spring Boot, allows us to:

- **Store Stack Overflow Data:** Persist questions, answers, and tags, not just as rows in tables, but as interconnected nodes.
- **Ask Compelling Questions:** Use graph query language (Cypher) to uncover patterns, find similar questions, or identify experts based on tags.
- **Build a Smarter Search:** Go beyond keyword matches and offer results that understand the relationships within Stack Overflow's knowledge base.

The Journey Continues

Lets fetch node, and then some property of it (links)

```
@SpringBootApplication
@ConditionalOnProperty(name = "runner.displayGraphNodes.enabled", havingValue = "true")
public class DisplayGraphNodes implements CommandLineRunner {

    1 usage
    @Autowired
    Driver driver;


    public static void main(String[] args) {
        System.out.println("Staring GraphDB App");
        SpringApplication.run(DisplayGraphNodes.class, args);
    }


    @Override
    public void run(String... args) throws Exception{
        try(var session= driver.session()){
            System.out.println("Staring GraphDB session..");
            session.run( s: "match (n) return n.link as link limit 30").list().forEach( r -> {
                    System.out.println(r.get("link"));
                }
            );
        }
    }
}
```

Key Components

Annotations

- `@SpringBootApplication` : This marks the class as your main Spring Boot application entry point.
- `@ConditionalOnProperty(name = "runner.displayGraphNodes.enabled", havingValue = "true")` : This crucial annotation controls whether the `CommandLineRunner` logic will execute. It checks if a property named "runner.displayGraphNodes.enabled" is set to "true" `application.properties`

Dependency

- `@Autowired Driver driver;` : This injects a `Driver` object (from the Neo4j Java driver) for interacting with your Neo4j database.

Main Method

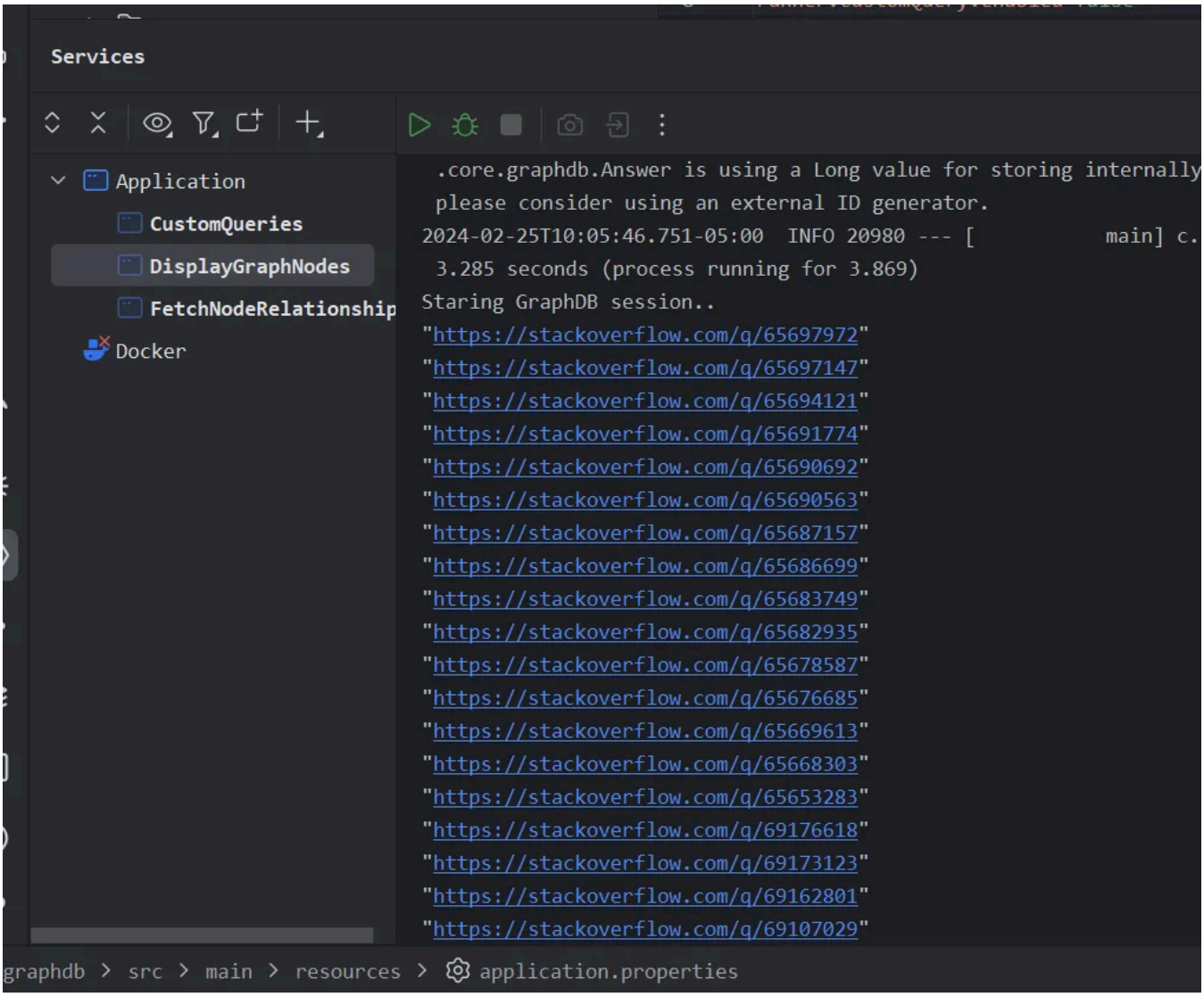
- `public static void main(String[] args) { ... }` : Typical Spring Boot application entry point. Here, it triggers the execution of your Spring application context.

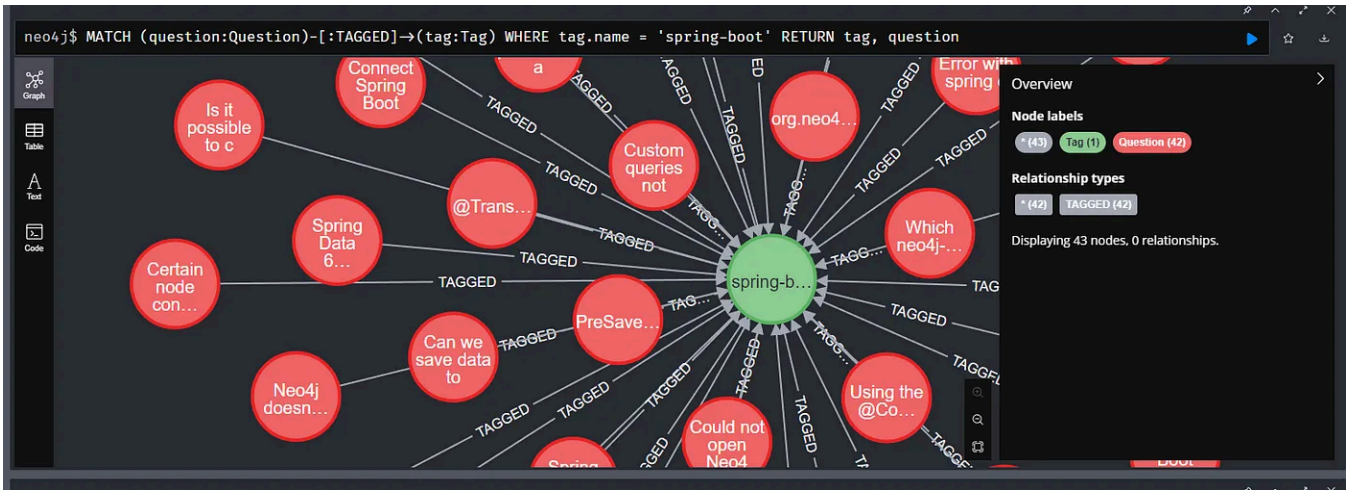
CommandLineRunner Implementation

- `public void run(String... args) throws Exception { ... }` : This is the core logic that will run if the conditional property is enabled.

fetches up to 30 nodes (assuming they have a 'link' property) and gets their 'link' value.

- Prints the retrieved 'link' values to the console.





Demonstrating Fetching Node Relationships

```java
@SpringBootApplication
@EnableNeo4jRepositories(considerNestedRepositories = true)
@ConditionalOnProperty(name = "runner.fetchNodeRelationship.enabled", havingValue = "true")
public class FetchNodeRelationship implements CommandLineRunner {

    no usages
    @Autowired
    Driver driver;

    1 usage
    @Autowired
    QuestionRepository questionRepository;

    public static void main(String[] args) {
        System.out.println("Staring GraphDB App");
        SpringApplication.run(FetchNodeRelationship.class, args);
    }

    @Override
    public void run(String... args) throws Exception {
        Question question = questionRepository.findById("How to save multiple labels of a node in neo4j in a specifi
        System.out.println("question is ::  " + question);
        question.tags.forEach(t -> System.out.println(t));
    }

    1 usage
    interface QuestionRepository extends Neo4jRepository<Question, String> {}
}
```

Let's dissect this code snippet and explore its purpose within your Spring Boot and Neo4j application.

The central goal of this code is to showcase how to retrieve a node from your Neo4j graph database and then navigate its relationships to access connected data. Here's a breakdown:

Key Components:

Annotations

- `@SpringBootApplication` : Marks your main Spring Boot application class.
- `@EnableNeo4jRepositories` : Enables Spring Data Neo4j repositories, simplifying your database interactions.
- `@ConditionalOnProperty` : Like in the previous example, this controls the execution of your `CommandLineRunner` logic based on a property setting.

Dependencies

- `@Autowired Driver driver;` : Injects a Neo4j `Driver` (likely not used directly in this particular snippet's logic).
- `@Autowired QuestionRepository questionRepository;` : Injects your `QuestionRepository` , providing the primary interaction mechanism with Neo4j.

Main Method

- Standard Spring Boot application entry point.

1. **CommandLineRunner Implementation**

- `@Override public void run(String... args)` : This code executes if the "runner.fetchNodeRelationship.enabled" property is set to "true".
- **Fetch Question:** `questionRepository.findById("How to save multiple labels of a node in neo4j in a specified order?").get();` retrieves a `Question` node using its title as the identifier
- **Print Question:** Prints the retrieved `Question` object.
- **Traverse Relationship:** `question.tags.forEach(t -> System.out.println(t))` demonstrates how to seamlessly access the associated `Tag` nodes connected via the "TAGGED" relationship defined in your `Question` model. Each `Tag` is printed.

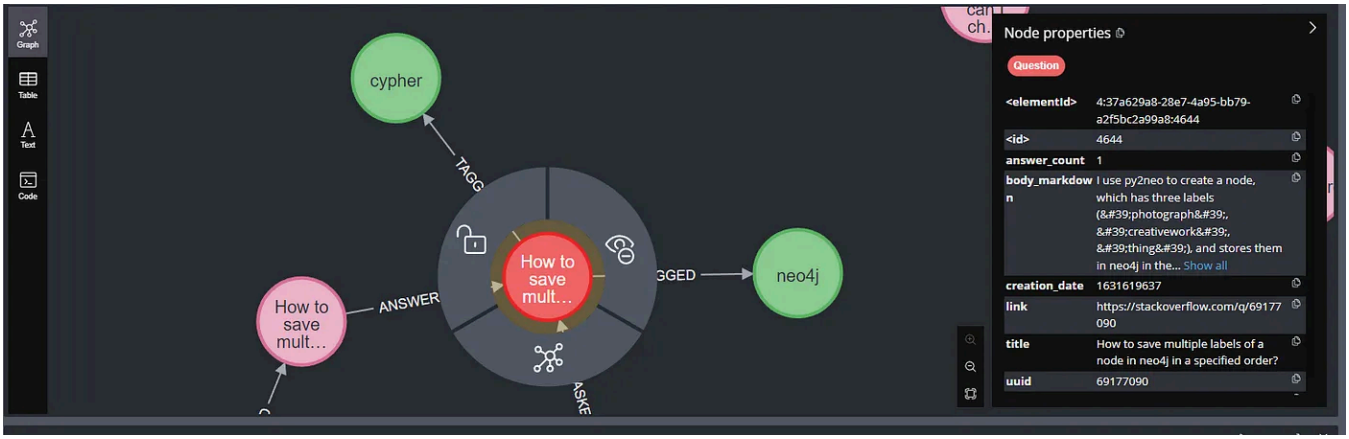
In Essence

This code highlights these concepts:

- **Spring Data Neo4j Repositories:** Using a repository interface to streamline fetching nodes.
- **Object-Graph Mapping (OGM):** Spring Data Neo4j automatically maps tags (related nodes) to the `tags` property in the `Question` object.
- **Power of Relationships:** It showcases how a graph database makes it simple to fetch connected data.

```
question is ::  Question[title='How to save multiple labels of a node in neo4j in a specified order?', quesWAnswerCount=1,
 tags=[Tag[name='cypher'], Tag[name='neo4j']]]
Tag[name='cypher']
Tag[name='neo4j']
2024-02-25T10:07:27.897-05:00  INFO 24756 --- [ionShutdownHook] o.neo4j.driver.internal.InternalDriver   : Closing driver instance
 804347788
2024-02-25T10:07:27.905-05:00  INFO 24756 --- [ionShutdownHook] o.n.d.i.async.pool.ConnectionPoolImpl    : Closing connection pool
 towards localhost:7687

Process finished with exit code 0
```



Filtering Questions by Answer Count

```java
@SpringBootApplication
@EnableNeo4jRepositories(considerNestedRepositories = true)
@ConditionalOnProperty(name = "runner.customQuery.enabled", havingValue = "true")
public class CustomQueries implements CommandLineRunner {

    no usages
    @Autowired
    Driver driver;

    1 usage
    @Autowired
    CustomQuestionRepository questionRepository;

    public static void main(String[] args) {
        System.out.println("Staring GraphDB App");
        SpringApplication.run(CustomQueries.class, args);
    }

    @Override
    public void run(String... args) throws Exception {
        List<Question> questions = questionRepository.getAllQuesWAnswers( ansCount: 3);
        questions.forEach(System.out::println);
    }

    1 usage
    interface CustomQuestionRepository extends Neo4jRepository<Question, String> {
        1 usage
        @Query("match (n:Question) where n.answer_count >= $ansCount return n")
        List<Question> getAllQuesWAnswers(@Param("ansCount") Integer ansCount );
    }
```

The primary function of this code is to demonstrate how to create a custom query method within a Spring Data Neo4j repository to retrieve data using specific criteria that go beyond the basic methods provided by repositories.

Key Components

Annotations

- `@SpringBootApplication`, `@EnableNeo4jRepositories` : Standard setup for a Spring Boot application with Neo4j integration.
- `@ConditionalOnProperty` : Controls the execution of your `CommandLineRunner` based on the "runner.customQuery.enabled" property.

Dependencies

- `@Autowired Driver driver;` : Injects a Neo4j `Driver` .

- `@Autowired CustomQuestionRepository questionRepository;` : Injects your custom repository, providing access to your query.

Main Method

- Standard Spring Boot application entry point.

CommandLineRunner Implementation

- `@Override public void run(String... args)` : This code executes if the conditional property is enabled.
- **Custom Query Execution:** `questionRepository.getAllQuesWAnswers(3);` calls your custom query method, filtering questions with an `answer_count` of at least 3.
- **Print Results:** The returned list of questions is printed.

CustomQuestionRepository

- **Interface Definition:** Declares the `CustomQuestionRepository` interface extending the `Neo4jRepository`.
- **Custom Query:**
- `@Query("match (n:Question) where n.answer_count >= $ansCount return n")` The Cypher query to fetch questions with the specified answer count criterion.
- `@Param("ansCount") Integer ansCount` : Maps the `ansCount` method parameter to the `$ansCount` parameter within the Cypher query.

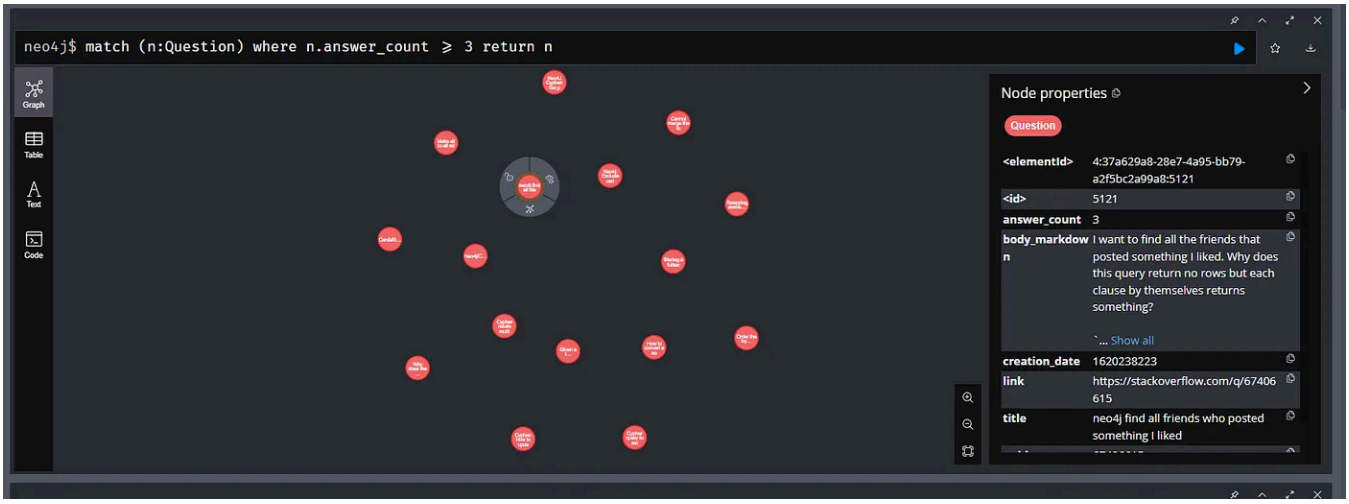
In Summary

This code demonstrates:

- **Custom Cypher Queries:** How to embed Cypher queries directly in your Spring Data Neo4j repository interface for tailored data retrieval.
- **Parameterized Queries:** Making queries more flexible by passing in values (e.g., the minimum answer count) at runtime.
- **Benefits of Repositories:** Custom queries streamline interactions with Neo4j, enhancing the standard functionality provided by the repository.

```
2024-02-25T10:11:15.150-05:00  INFO 16244 --- [           main] c.a.test.core.graphdb.CustomQueries      : Started CustomQueries in 3.443
  seconds (process running for 4.052)
Question[title='Given a plain text and a possible cipher text, determine whether the cipher text can be formed from the plain text using
 the mentioned scheme', quesWAnswerCount=3, tags=[]]
Question[title='Cypher query to recursively traverse family tree?', quesWAnswerCount=3, tags=[]]
Question[title='How to convert a node to multiple relationships in Neo4j', quesWAnswerCount=3, tags=[]]
Question[title='Order the result by the number of relationships', quesWAnswerCount=3, tags=[]]
Question[title='filtering a fulltext search with a relationship constraint', quesWAnswerCount=3, tags=[]]
Question[title='Swapping elements in a list based on their match in a dictionary', quesWAnswerCount=3, tags=[]]
Question[title='Cannot merge the following node because of null property value for &#39;name&#39;:', quesWAnswerCount=4, tags=[]]
Question[title='Neo4j: Exclude certain nodes in variable path relationship', quesWAnswerCount=3, tags=[]]
Question[title='Neo4J Cypher: Set property only if it is null but still return the node', quesWAnswerCount=3, tags=[]]
Question[title='neo4j find all friends who posted something I liked', quesWAnswerCount=3, tags=[]]
Question[title='Make all to all relationship among the element present in list in neo4j', quesWAnswerCount=3, tags=[]]
Question[title='Can&#39;t use the added function in Repository', quesWAnswerCount=3, tags=[]]
Question[title='Neo4j/Cypher Import a CSV file', quesWAnswerCount=3, tags=[]]
Question[title='Why does the query to find intermediate nodes take so long?', quesWAnswerCount=3, tags=[]]
Question[title='Cypher return multiple hops through pattern of relationships and nodes', quesWAnswerCount=3, tags=[]]
Question[title='Cypher: how to update a property value in a node that exists in the Neo4j database?', quesWAnswerCount=3, tags=[]]
2024-02-25T10:11:15.727-05:00  INFO 16244 --- [ionShutdownHook] o.neo4j.driver.internal.InternalDriver   : Closing driver instance
 1048842522
2024-02-25T10:11:15.735-05:00  INFO 16244 --- [ionShutdownHook] o.n.d.i.async.pool.ConnectionPoolImpl    : Closing connection pool
 towards localhost:7687

Process finished with exit code 0
```



The true value of knowledge graphs lies in their ability to reveal insights that are difficult to see in traditional tabular data. I encourage you to continue experimenting and pushing the boundaries of what you can discover within your Stack Overflow or any knowledge graph!

Graph Database | Neo4j | Spring Boot | Machine Learning | AI

Follow

Written by Aniket Hingane | Day Manager, Night Coder

32 Followers

Passionate about simplifying software concepts and design through concise articles. Making complexity accessible, one short piece at a time.

More from Aniket Hingane | Day Manager, Night Coder

 Aniket Hingane | Day Manager, Night Coder

LLM apps : Why you MUST know Semantic Router in 2024 : Part 1

Must know LLM apps decision making layer, don't miss out !

· 6 min read · Jan 23, 2024

 -- 

Aniket Hingane | Day Manager, Night Coder

LLM Apps : Why Knowledge Graphs are super critical to know if you care about RAG : 1

AI Apps Need RAG, RAG Need Knowledge Graph ! Next part we will code RAG on KG.

· 5 min read · Feb 21, 2024

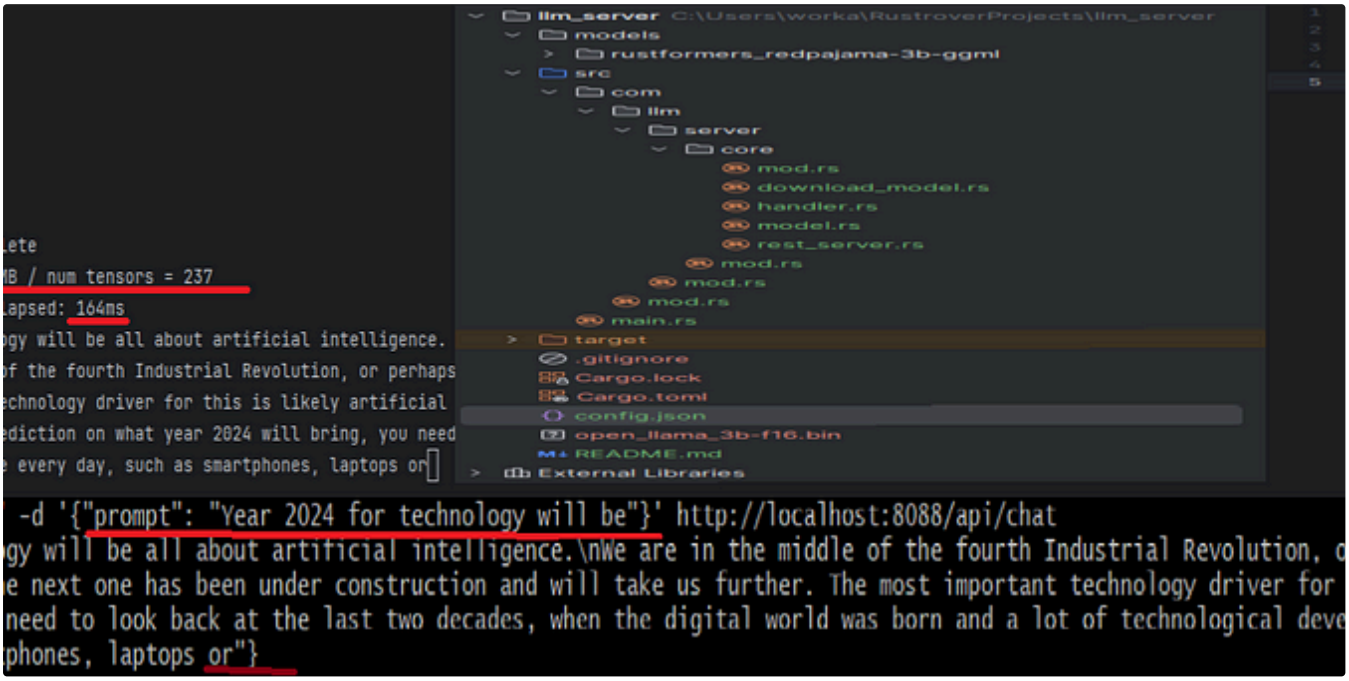
Aniket Hingane | Day Manager, Night Coder

LLM apps : Why you MUST know Semantic Router in 2024 : Part 2

Leveraging Semantic Router for Enhanced Prompting in Your Applications

✦ · 5 min read · Jan 24, 2024

--



Aniket Hingane | Day Manager, Night Coder

Must know for AI App Dev : Let's serve LLMs via Rust programing : LLM Handler Build

Let's build LLM handler and run super fast inference!

✦ · 5 min read · Feb 18, 2024

--

See all from Aniket Hingane | Day Manager, Night Coder

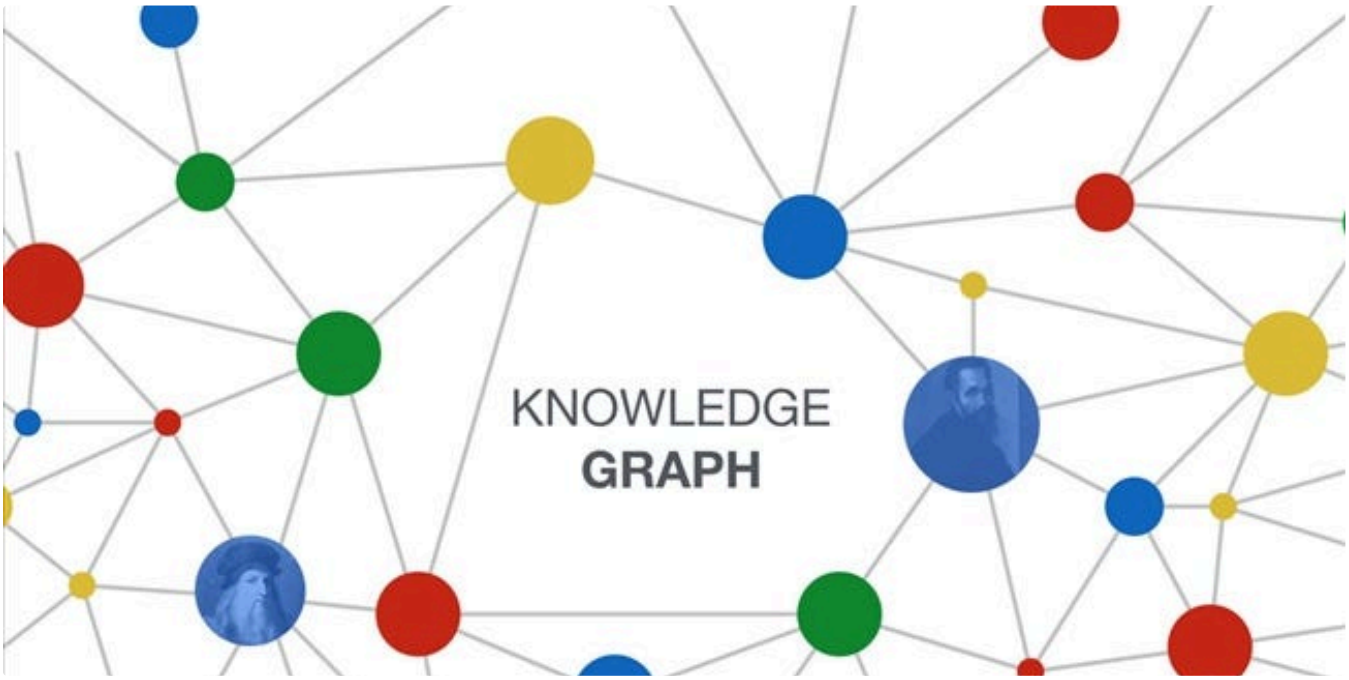
Recommended from Medium

Aniket Hingane | Day Manager, Night Coder

LLM Apps : Why Knowledge Graphs are super critical to know if you care about RAG : 1

AI Apps Need RAG, RAG Need Knowledge Graph ! Next part we will code RAG on KG.

· 5 min read · Feb 21, 2024



Phil

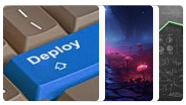
Knowledge Graphs in RAGs (with Llama-Index)

Are #KnowledgeGraphs (KGs) better than #VectorDBs in RAGs? Yes and No!

3 min read · Jan 29, 2024

--

Lists



Predictive Modeling w/ Python

20 stories · 951 saves

The New Chatbots: ChatGPT, Bard, and Beyond

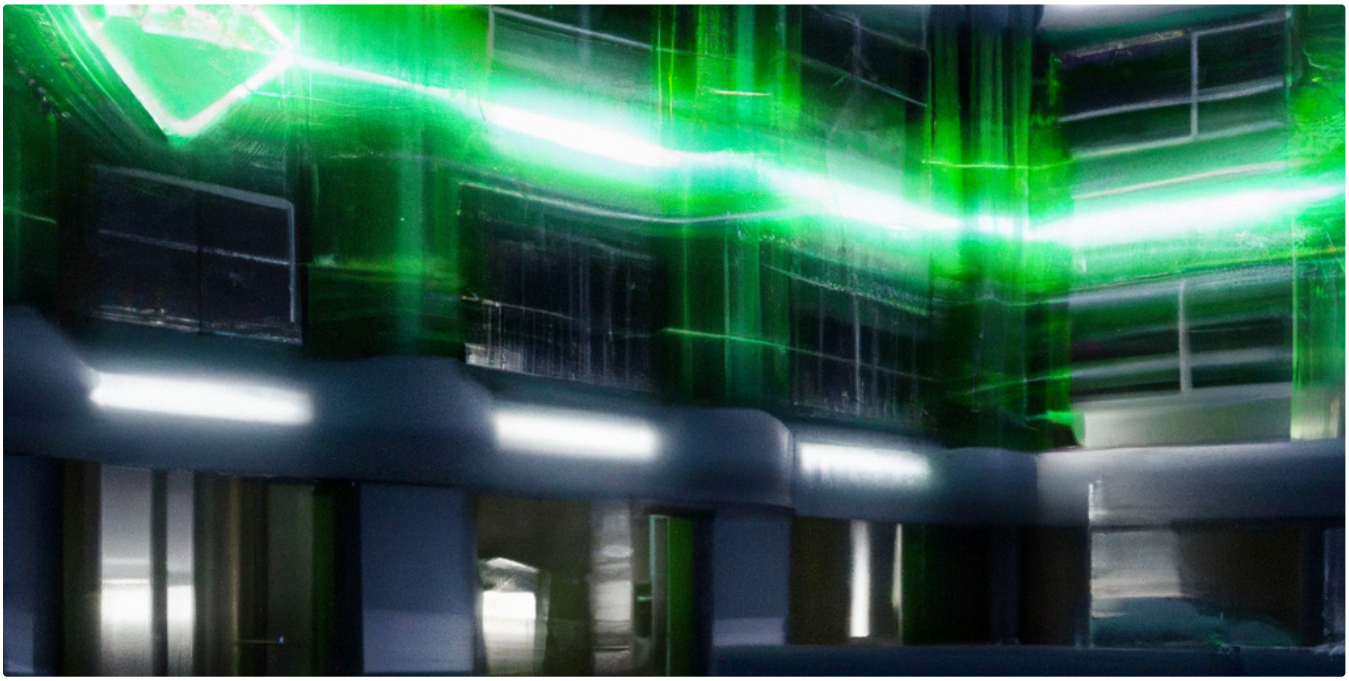
12 stories · 318 saves

Practical Guides to Machine Learning

10 stories · 1121 saves

Natural Language Processing

1232 stories · 719 saves

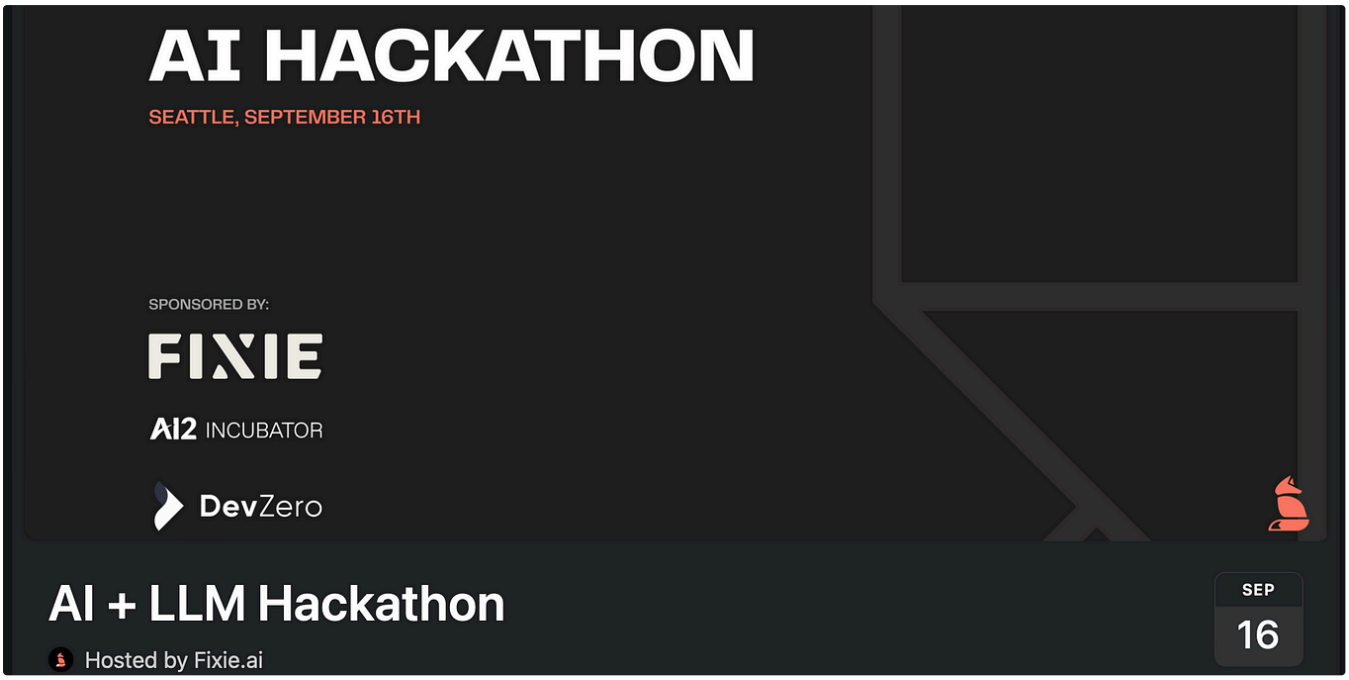


AIWorldBlog

GraphRAG: Enhancing AI Understanding of Unseen Data

2 min read · Feb 13, 2024

-- 1



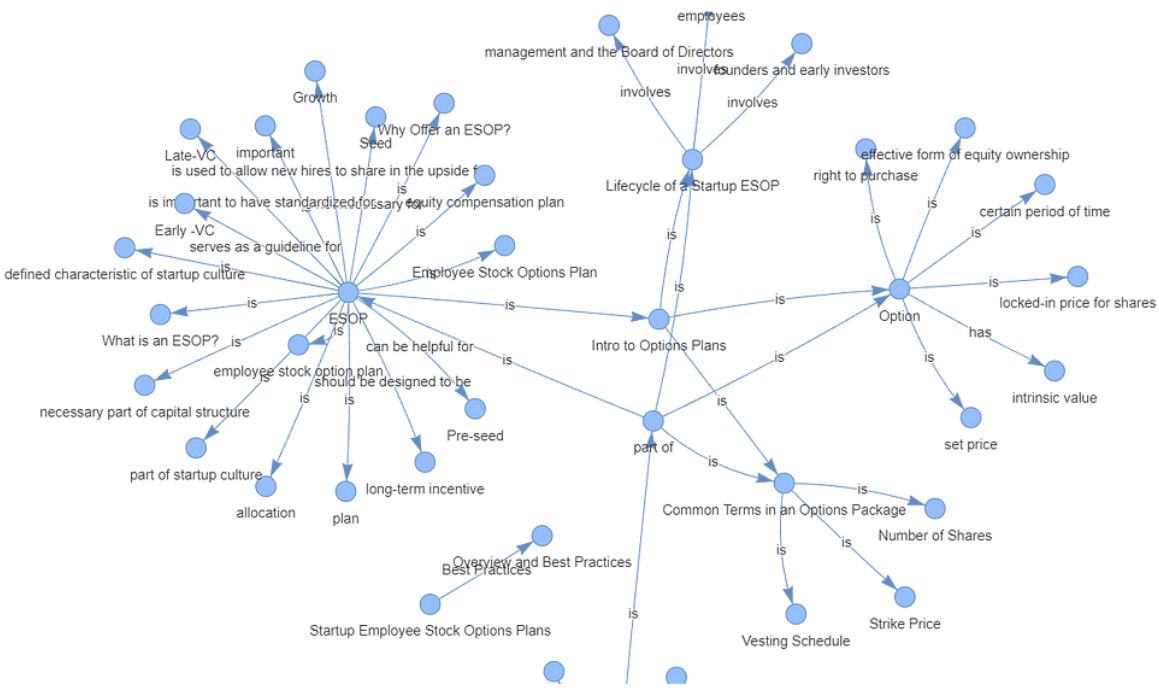
Rob Brennan

Explore OpenAI vector embedding with Neo4j, LangChain, and Wikipedia

I'm a huge Neo4j fan. Since being introduced to graph databases, I've always had an ear out for exploring intriguing use cases. Over my...

22 min read · Sep 18, 2023

-- 2



Plaban Nayak in AI Planet

Implement RAG with Knowledge Graph and Llama-Index

Hallucination is a common problem when working with large language models (LLMs). LLMs generate fluent and coherent text but often generate...

25 min read · Dec 3, 2023

-- 7

NebulaGraph Database

Graph RAG: Unleashing the Power of Knowledge Graphs with LLM

In the era of information overload, sifting through vast amounts of data to provide accurate search results in an engaging and...

6 min read · Sep 8, 2023

-- 4

See more recommendations