AI Foundation

Knowledge Graph

Introduction to Knowledge Graph and its applications

Sept 30, 2025

LESSON PLAN

01 **Introduction to Knowledge Graph**

- Components

02 **GraphRAG vs. RAG**

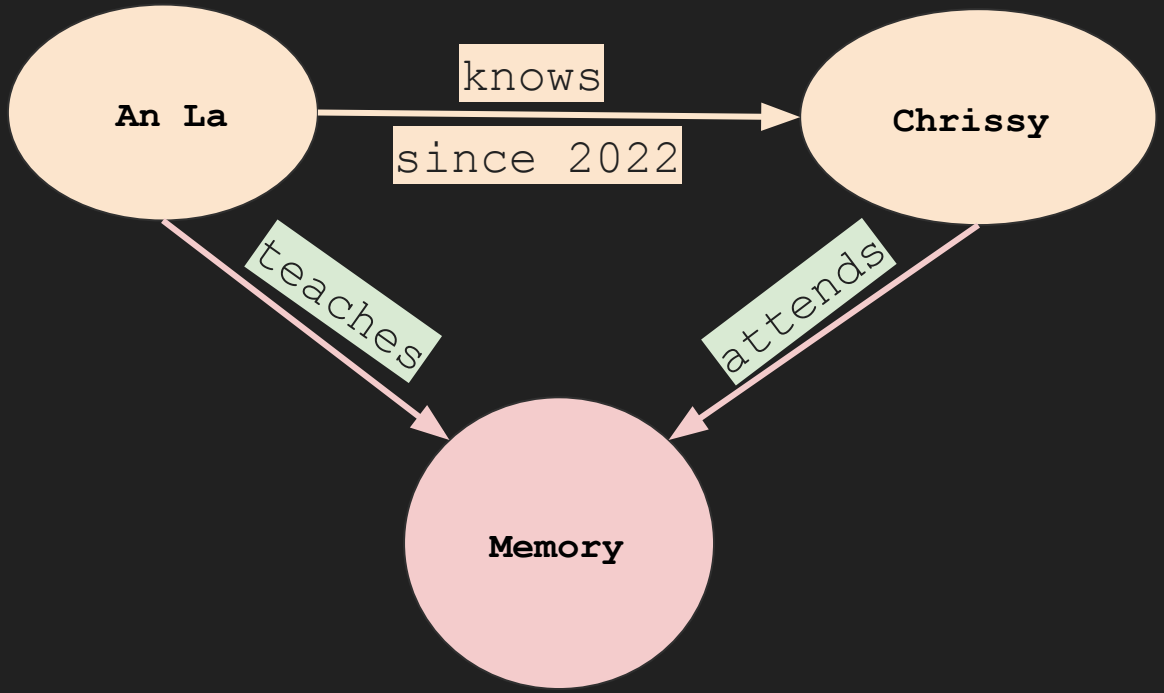
- Limitation on RAG
- Main differences: GraphRAG vs. RAG

03 **GraphRAG - Under the hood**

04 **How to build a Knowledge Graph**

Define Knowledge Graph

Example

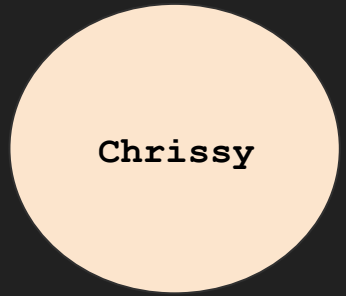


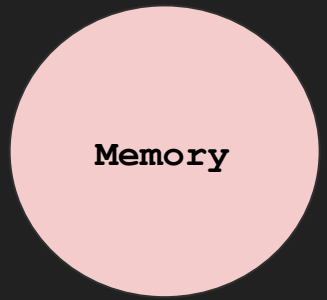

```
(Person)-[Teaches]→ (Session) ←[Attends]- (Person)
```

Component #1: Node

Node (Entity)

- a data record
- represents entities (people, thing, concept, etc)
- has label, e.g. **Person**
- key-value property:
  - **name "Chrissy"**



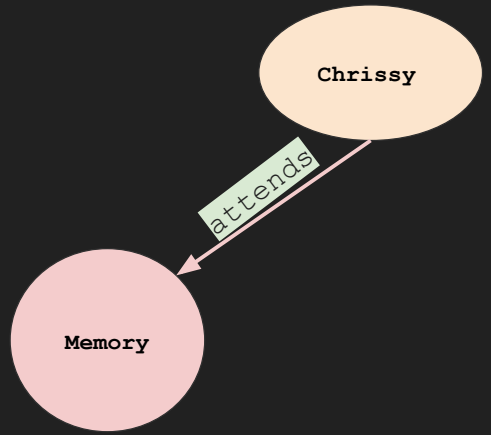


Component #2: Edge

Edge (Relationship)

- represents relationship
- connection b/w two nodes
- **has one direction**
- source vs. target node

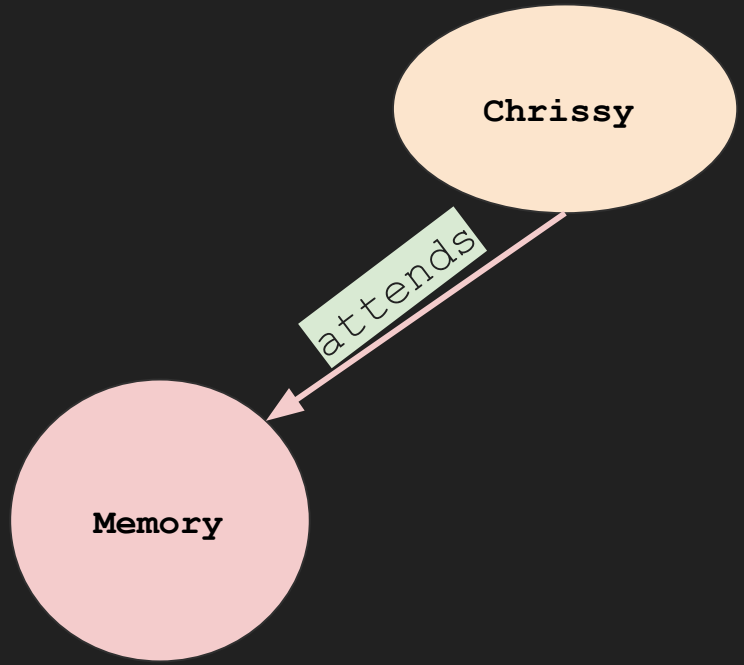
- **Taxonomic**: `:IS_A`, `:SUBCLASS_OF` (usually transitive)
- **Part–whole**: `:PART_OF`, `:HAS_PART` (often transitive)
- **Membership**: `:MEMBER_OF`, `:BELONGS_TO`
- **Attribution / property-as-edge**: `:HAS_ATTRIBUTE`, `:HAS_STATUS` (or use DatatypeProperty in RDF)
- **Identity / equivalence**: `:SAME_AS` (e.g., `owl:sameAs`)
- **Causal / dependency**: `:CAUSES`, `:REQUIRES`, `:DEPENDS_ON`
- **Temporal**: `:VALID_DURING`, `:HAPPENED_BEFORE/AFTER` (or put timestamps on the edge)
- **Spatial**: `:LOCATED_IN`, `:NEAR`
- **Event roles** (n-ary relations via an event node): `(:Event)<-[:AGENT]-(:Actor)`, `(:Event)-[:PATIENT]->(:Asset)`
- **Provenance**: `:CREATED_BY`, `:DERIVED_FROM`, `:CITED_BY`
- **Similarity / relatedness**: `:SIMILAR_TO` (often with a `score` / `weight`)



Component #3: Properties

Properties

- additional info about nodes & edges
- Data types: string, integer, float, array, boolean, etc



Knowledge graph is..

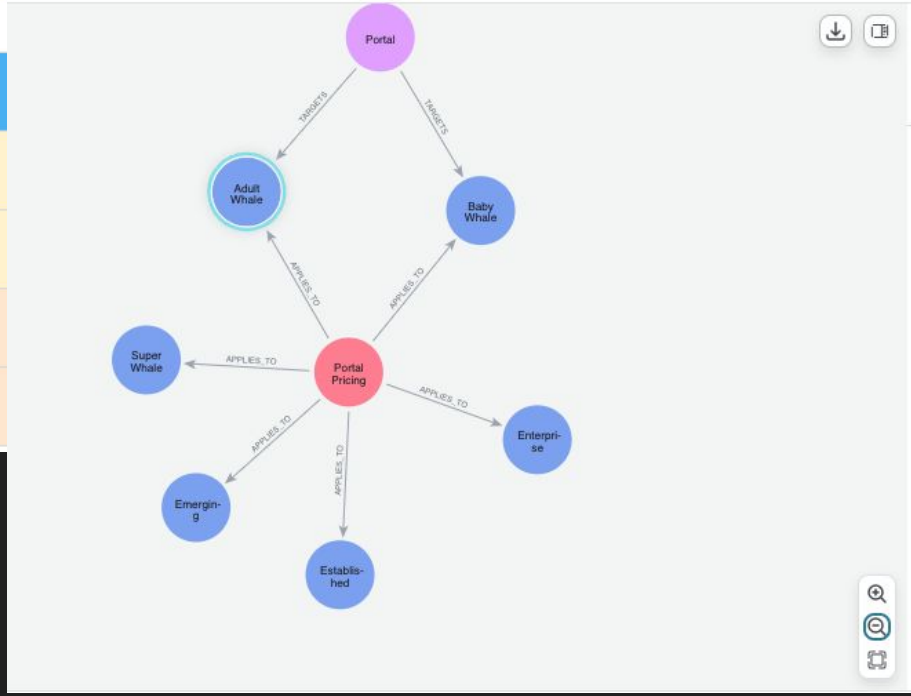
- A database that stores information in **nodes** and **edges**.
  - Both nodes and edges have **properties**.
  - Edges have **direction** and **types**.

(Source: Deep Learning AI - Knowledge Graph for RAG)

Example on Neo4j

Proposed Portal Pricing

Fee Type	Unit
Annual Fee (Recurring)	Per-Fund
	Per-SPV
Data Migration Fee (One-time)	Per-Fund
	Per-SPV



Node details

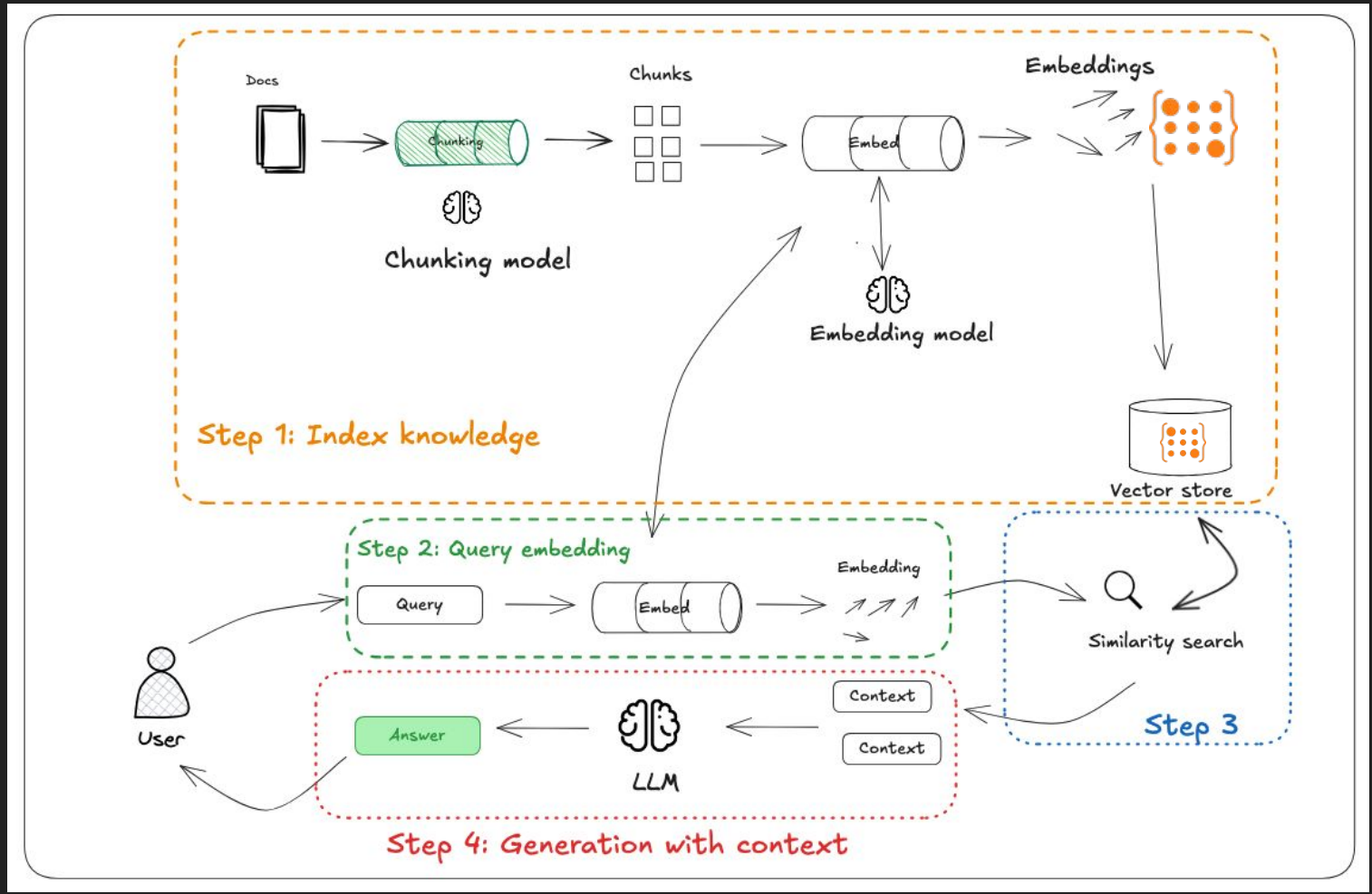
CustomerSegment

Key	Value
<id>	4:334e6844-58de-4985-bfcb-c60dff69e729:7
annual_fund_fee	20000
annual_spv_fee	4000
aum_range	"$50-100B"
migration_fund_fee	7500
migration_spv_fee	1500
name	"Adult Whale"

Deep dive into Knowledge Graph setup

The idea of GraphRAG and more

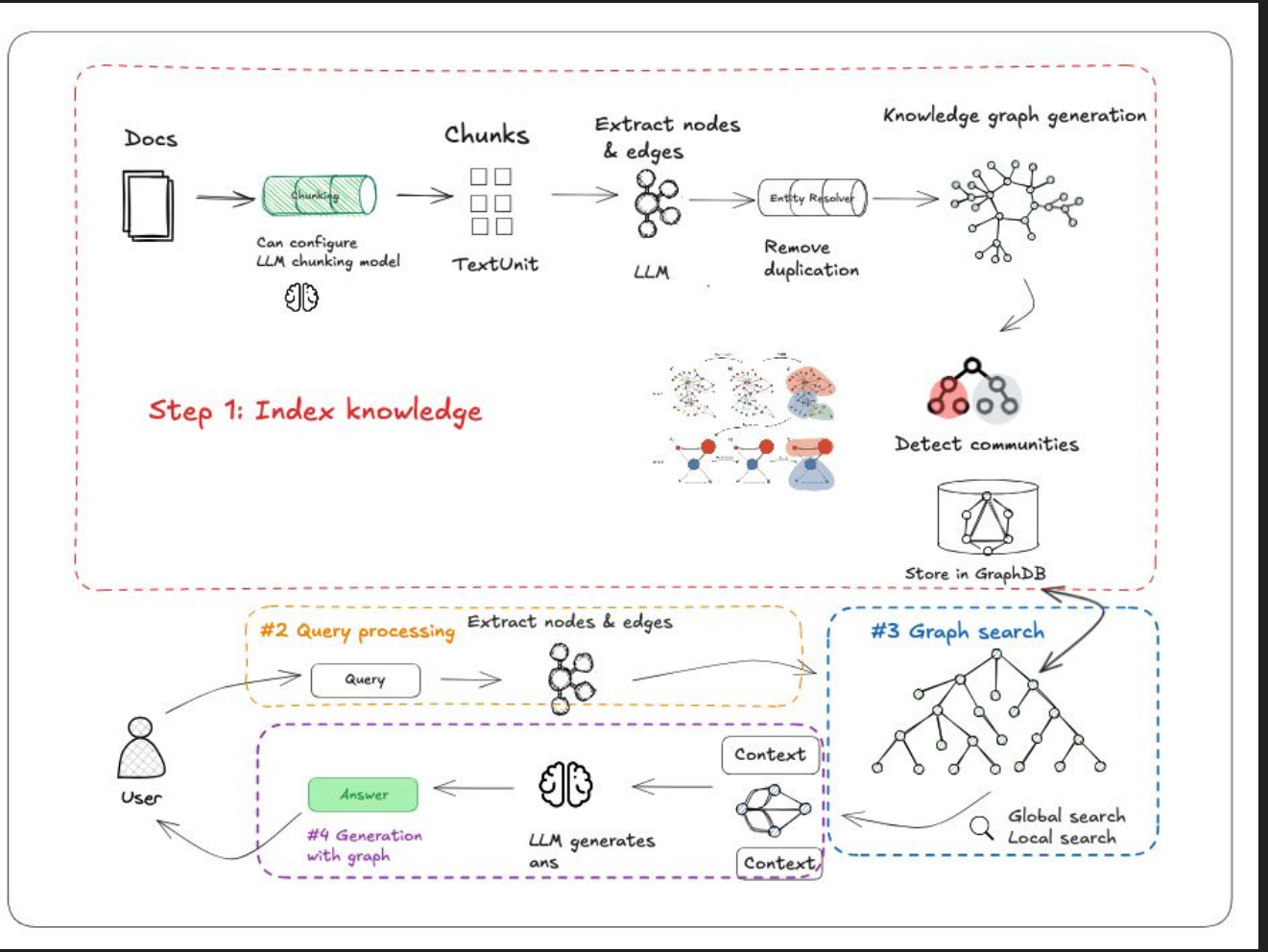
RAG


Docs
Chunking
Chunks
Embed
Embeddings
Chunking model
Embedding model
Step 1: Index knowledge
Vector store
Step 2: Query embedding
Query
Embed
Embedding
Similarity search
Step 3
User
Answer
LLM
Context
Context
Step 4: Generation with context

When RAG fails to deliver

- Simple keyword/semantic search may miss complex relationships
- Difficulty with multi-hop reasoning
- Limited context understanding across document boundaries
- Challenges with entity disambiguation

Here comes Graph RAG

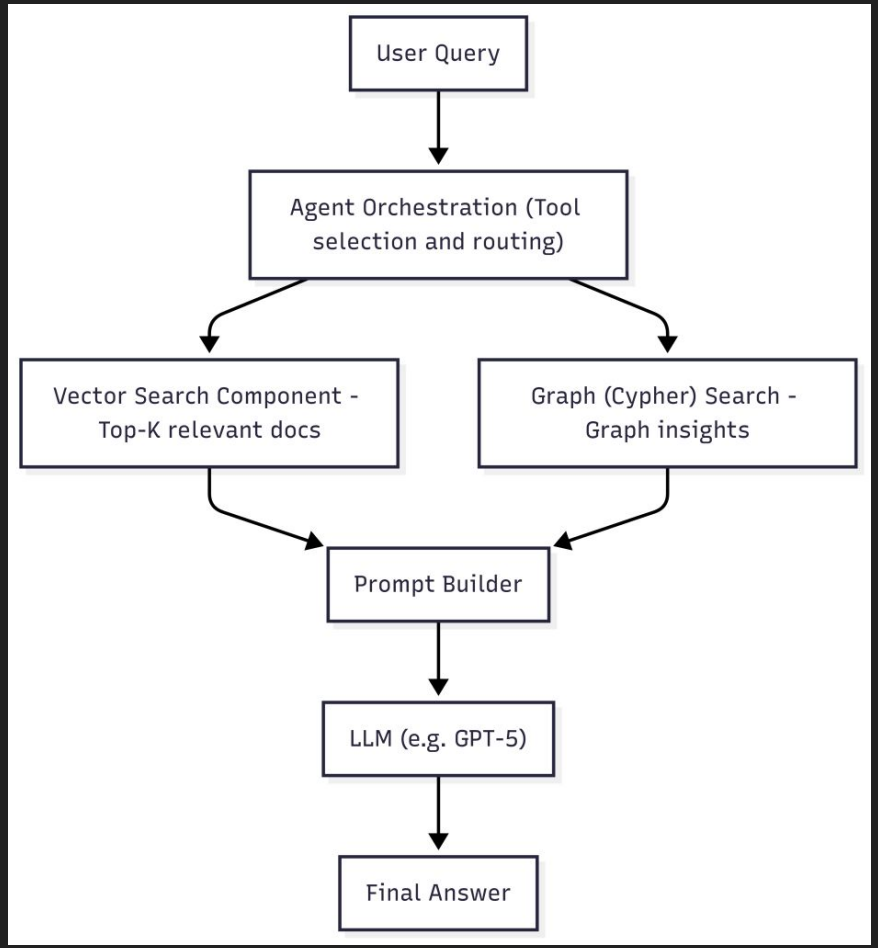


For reference

GraphRAG architecture

This combination of a knowledge graph and vector search is particularly effective when your application requires:

- Reasoning over complex architectures (e.g., microservices, IT assets, workflows)
- Querying both structured metadata and unstructured documentation
- Combining multiple sources into one coherent system



Compare RAG vs. GraphRAG

Aspect	Traditional RAG	GraphRAG
Structure	Flat vector search	Graph-based relationships
Context	Individual chunks	Connected knowledge network
Reasoning	Similarity-based	Relationship-based
Multi-hop	Limited	Excellent
Complexity	Lower	Higher

GraphRAG Glossary

Text Unit - a chunk of text. The size of these chunks, their overlap can be configured.

NER - Named-Entity-Recognition

Communities - a group of nodes that are more densely connected to each other than to nodes outside the group.

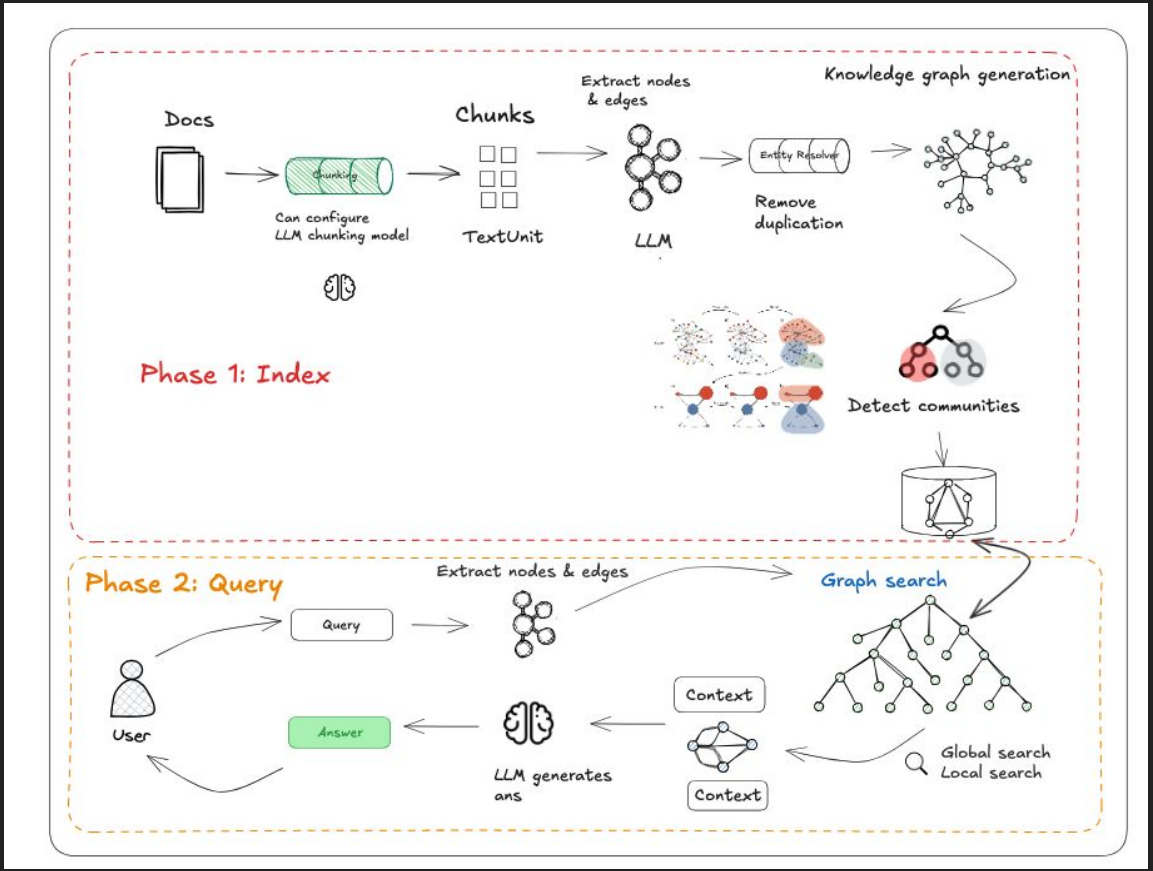
Community Report - The contents of each community are summarized into a generated report, useful for human reading and downstream search.

Community level - the hierarchy tier of clustering; level 0 is the most generic grouping, and higher numbers mean finer, more specific sub-communities.

Leiden - algorithm used to perform community detection.

GraphRAG - Under the hood

What's included in a GraphRAG?



Phase #1: Indexing - Graph Generation

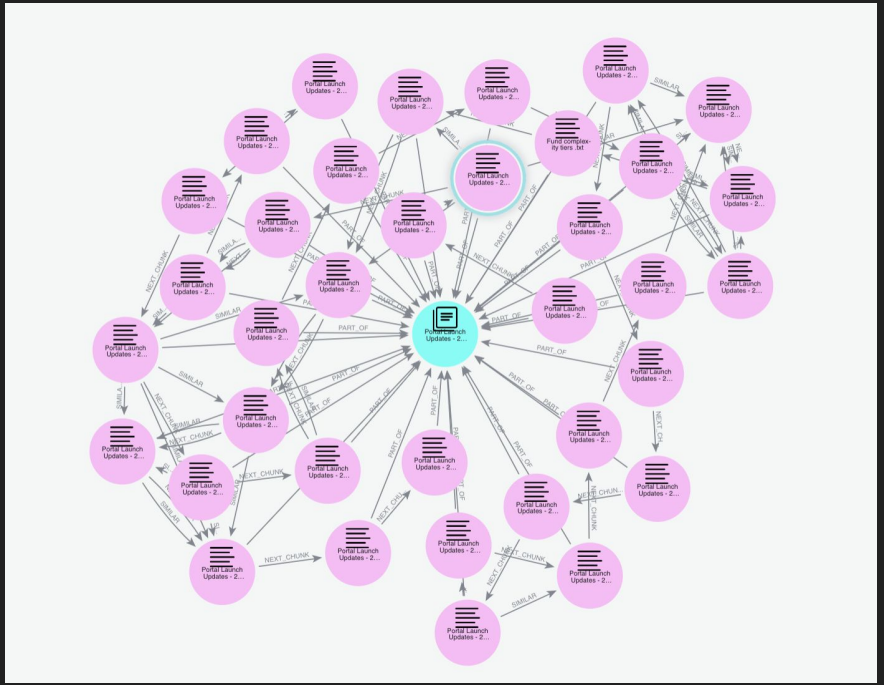
In this phase, there are a few steps:

1. Document → **Chunks**
2. LLM **extracts** entities + relationships from each chunk
3. Merge overlapping entities across chunks (entity resolution)
4. Build knowledge graph from consolidated entities/relationships
5. Apply Leiden algorithm → Detect hierarchical communities
6. Generate community summaries at multiple levels (bottom-up)
7. Create community reports describing each cluster's themes

#1 Text chunking

There are multiple chunking strategies:

- **Splitting:** (common)
  - Split a document into equal sized sections (by character or token-count),
  - with an optional overlap (typical sizes are 250-500 tokens with 50-100 tokens overlap)
- **Hierarchical Document Chunking:**
  - Split a document alongside lexical boundaries - chapters, sections, paragraphs
- **Sentence Chunking:** Split a document into individual sentences



Source: Neo4j

Why chunking?

Why chunking?

- Still need to chunk so LLM does not need to handle all at once

Is chunking same as RAG?

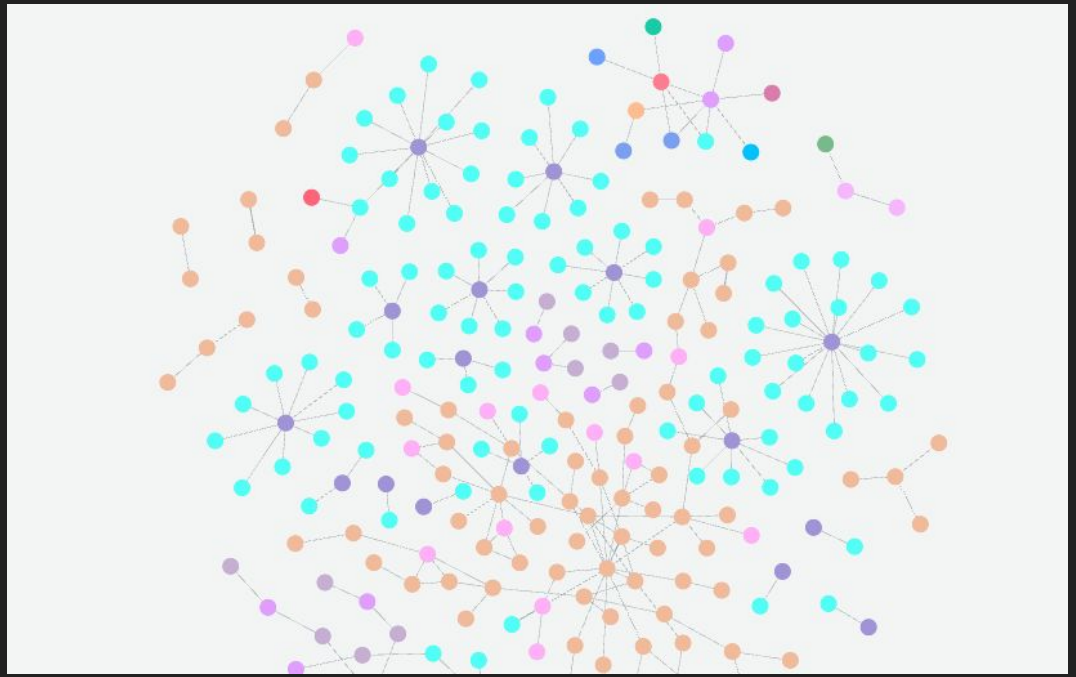
- Same chunking but embedding stored in vector database (optional)

What is the output after chunking?

- Text Units then go through entity extraction process.

#2 Graph extracting

- After chunking, next we will extract nodes and edges using LLM
- Apply Named-entity recognition (NER) to recognize entities and remove duplication
- The default graph shape is lexical graph*

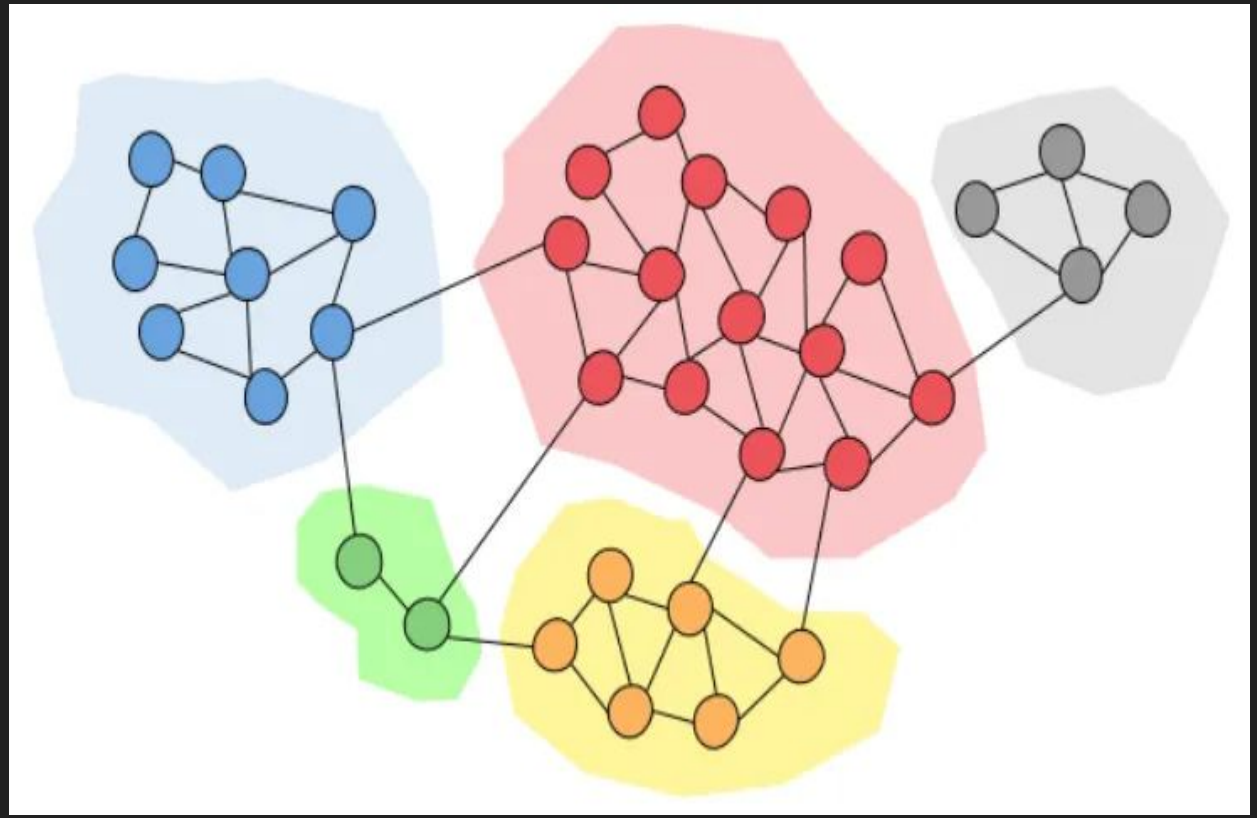


Source: Neo4j

#3 Community detection, then generate community reports (optional)

- Apply Hierarchical Leiden Algorithm to detect communities
- Generate summaries -> report

(source)



#4 Store in graph database

Database information

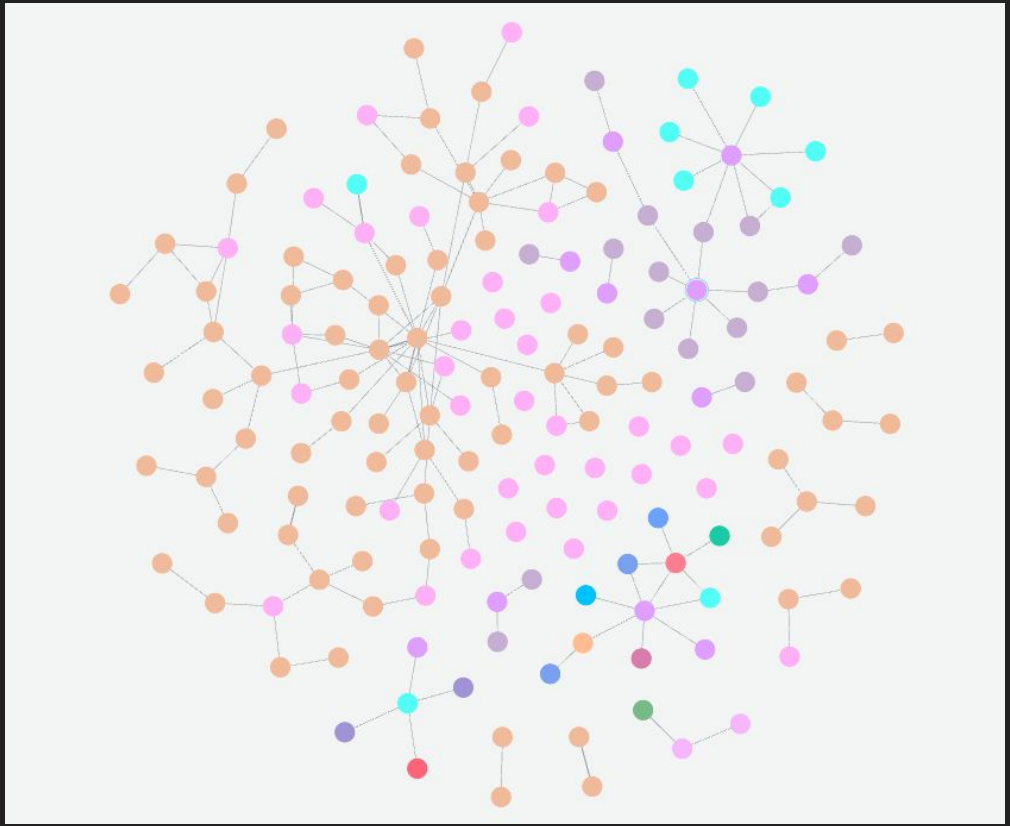
Nodes (430)

* __Entity__ Account AUMRange Campaign Characteristic Chunk Company Competitor CustomerGroup CustomerSegment DiscountStructure Document DocumentType EntityType Event Feature Fee FeeType FinancialEntity

Show all (26 more)

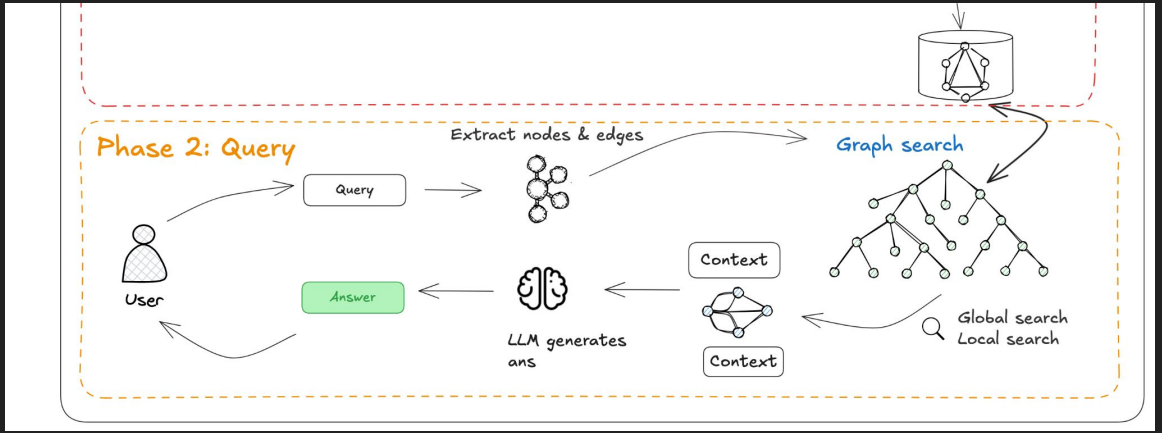
Relationships (1,206)

* ADD_ON_TO ALSO_KNOWN_AS APPLIES_TO APPLIES_TO_ENTITY ASSOCIATED_WITH belong_to_subcategory BELONGS_TO BETTER_THAN BUNDLED_WITH CAN_BUNDLE_WITH COMMUNICATES_TO COMPETES_WITH CONSISTS_OF CONSULTED_BY CONTAINS COORDINATES_ONBOARDING_FOR CREATES_LANDING_PAGE_FOR CREATES_PRESS_RELEASES_FOR DESCRIBES



Phase #2: Query

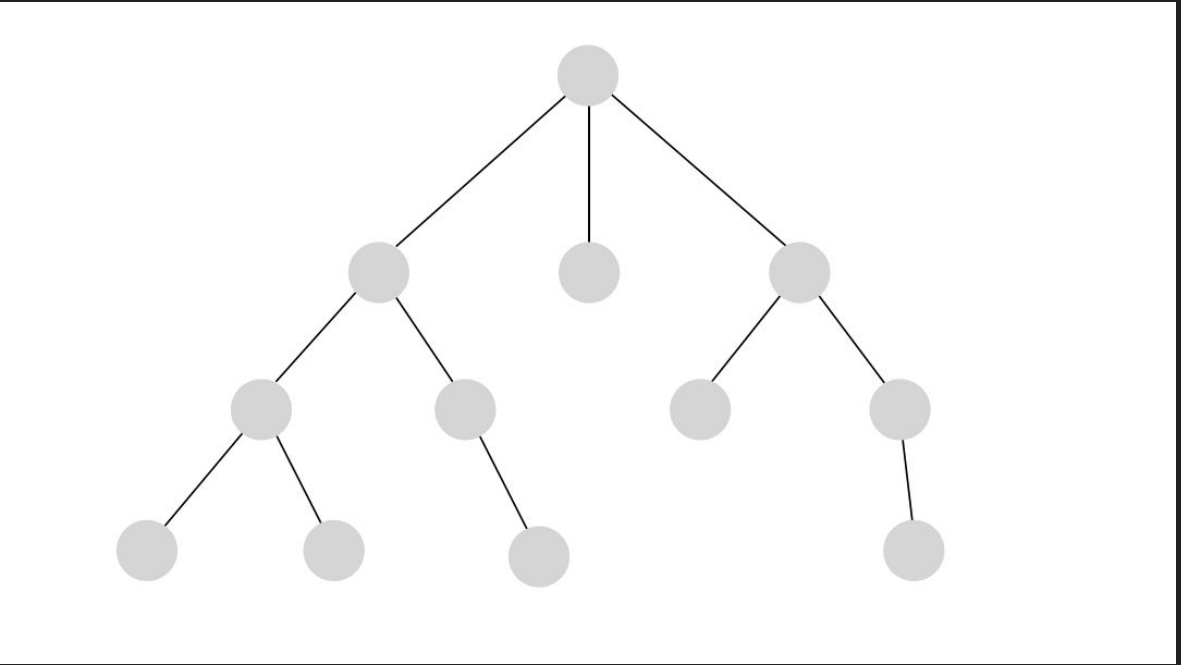
1. Extract query entities (using LLM)
2. Perform graph retrieval techniques:
   a. Entity-based
   b. Community-based
   c. Text-to-Cypher
   d. Others
3. Generate final response using retrieved context



More techniques can be viewed here

For entity-based retrieval

Graph traversal algorithms (methods like breadth-first search aka BFS): explores a graph by visiting all the neighboring nodes of a given level before moving to the next level



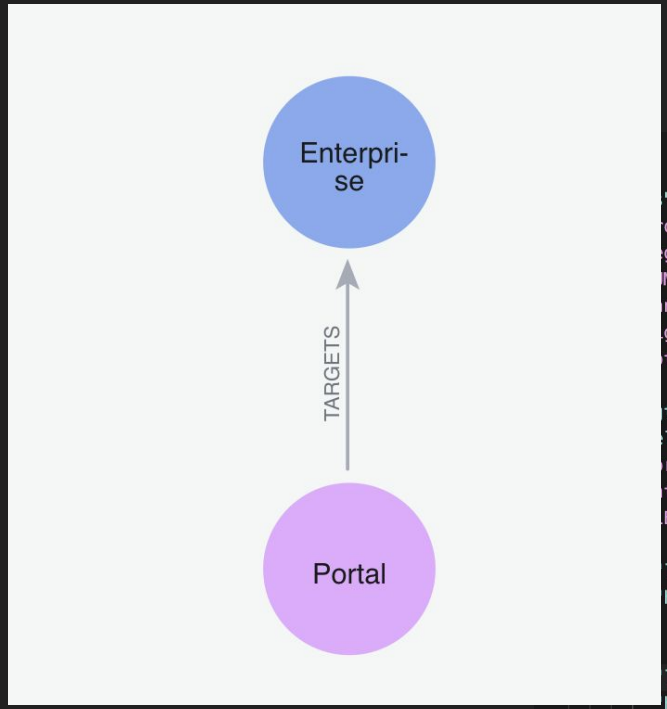
Text-to-Cypher

Query: What is the portal pricing for enterprise segment for funds?

```
1 MATCH (portal:Product {name: 'Portal'})-[:TARGETS]->(segment:CustomerSegment {name:
  'Enterprise'})
2 RETURN
3   portal.name AS Product,
4   segment.name AS Segment,
5   segment.aum_range AS AUM_Range,
6   segment.annual_fund_fee AS Annual_Fund_Fee,
7   segment.migration_fund_fee AS Migration_Fund_Fee,
8   (segment.annual_fund_fee + segment.migration_fund_fee) AS Total_First_Year_Cost
```

Product	Segment	AUM_Range	Annual_Fund_Fee	Migration_Fund_F	Total_First_Year
"Portal"	"Enterprise"	"$1B-15B"	15000	7500	22500

Output after graph search



Node details

CustomerSegment

Key	Value
<id>	4:334e6844-58de-4985-bfcb-c60dff69e729:5
annual_fund_fee	15000
annual_spv_fee	3000
aum_range	"$1B-15B"
migration_fund_fee	7500
migration_spv_fee	1500
name	"Enterprise"

Community-based retrieval - global search (source)

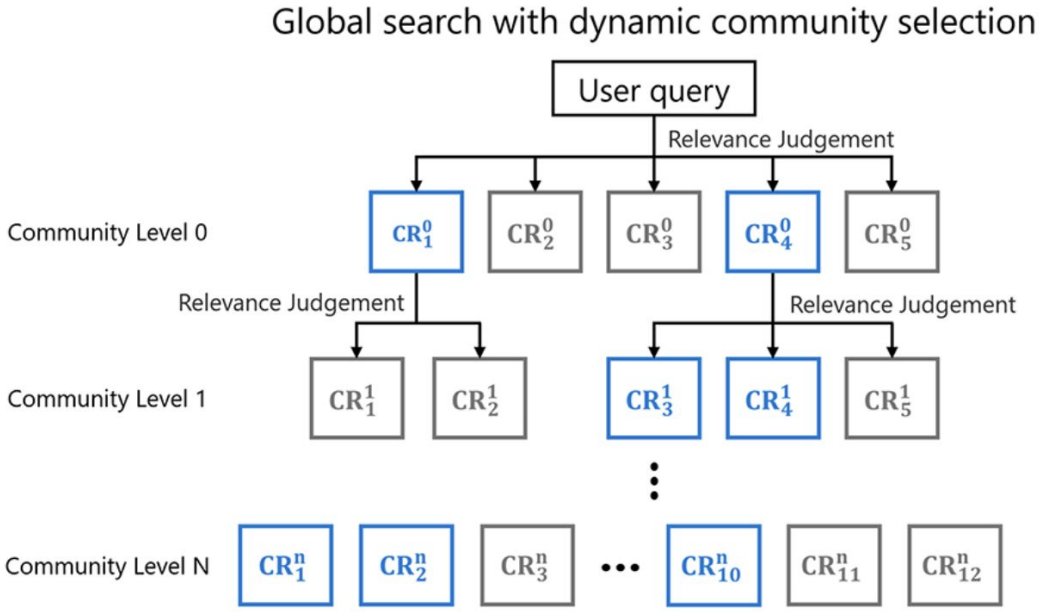


Figure 1: Dynamic community selection workflow

Step 1: Recursive Community Selection
Start at root level (level 0), using an LLM to rate the relevance of each community report (CR) against the user query.

If the community report is not relevant (grey boxes), then we move on to the next report.

If the community report is relevant, then we repeat the same operation to its child nodes. We keep the report (blue boxes) if none of its child communities is relevant or if it is a leaf node.

Step 2: Map-reduce Search
Perform map-reduce search on the selected community reports to generate the final response.

GraphRAG Glossary - p.2

Global search - handle queries that span the entire dataset

Local search - optimize for targeted queries , drawing from smaller subset of documents that closely match the user's input

DRIFT search - a hybrid b/w global and local search

Breadth-first search (BFS): explores a graph by visiting all the neighboring nodes of a given level before moving to the next level

Cypher — a graph-based query language developed by Neo4j

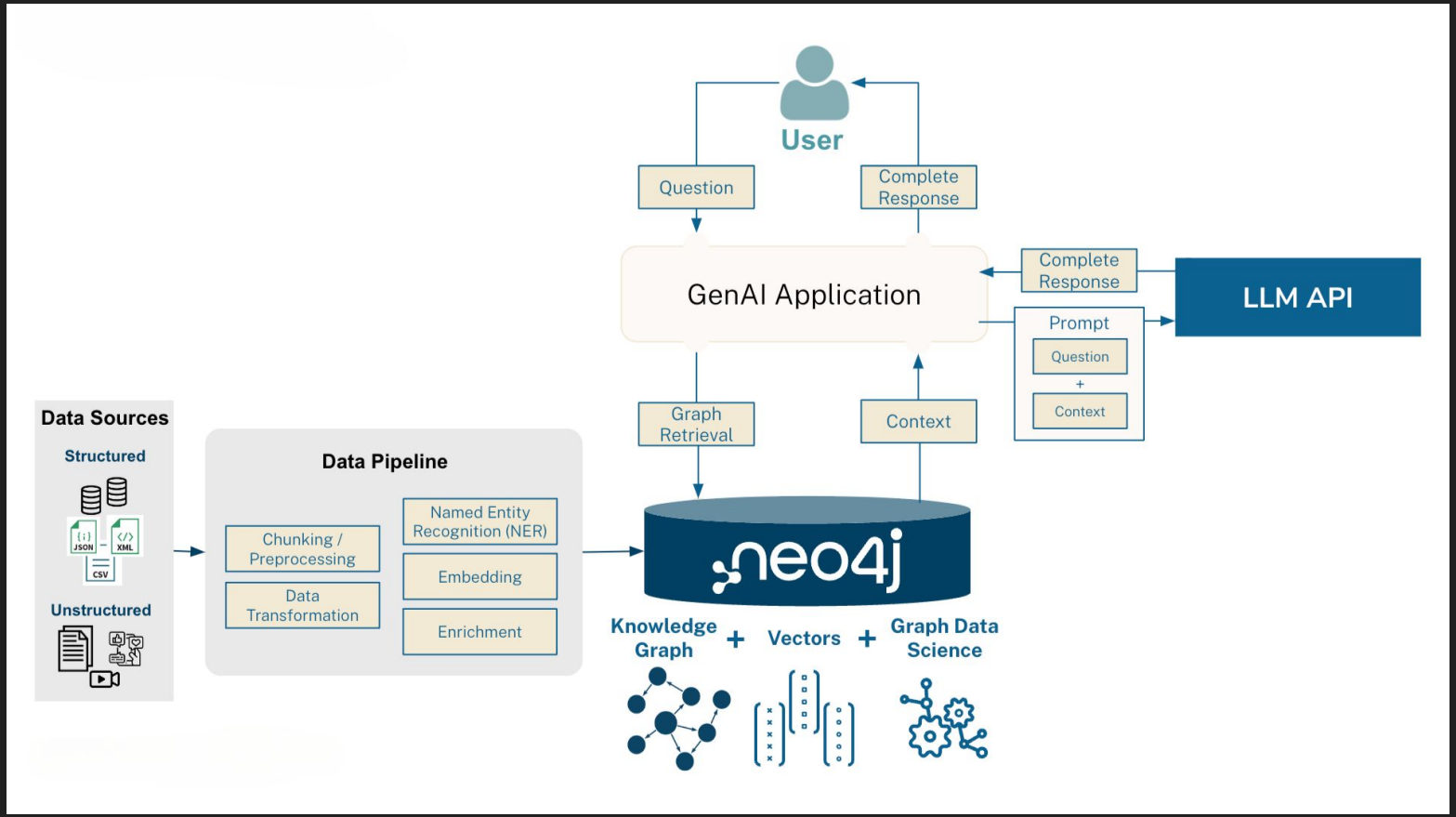
Example on Neo4j

Direct Answer: The Portal pricing for the **Enterprise** segment is composed of an **Annual Fund Fee** of **$15,000** and a **Migration Fund Fee** of **$7,500**, resulting in a **Total Pricing** of **$22,500**.

Key Insights:

- The total cost for the **Enterprise** segment stands at **$22,500** annually.
- The **Annual Fund Fee** constitutes approximately **66.67%** of the total pricing, while the **Migration Fund Fee** accounts for **33.33%**.

Graph RAG Neo4j



Part 3: How to build Knowledge Graph

Use cases & implementations

How-to

Step 1: Prepare data: structured and unstructured
Step 2: Configure the following items:

- chunking technique: sentence, semantic or else
- embedding model
- LLM used to generate graph: LlamaIndex, Langchain

Step 3: Determine the database

- DB for vector
- DB for graph: GraphRAG, Neo4j, Memgraph, LightRAG
- Hybrid: Neo4j

Demo workflow (Flowise)

Link

Demo workflow (ZEP)

- Store long-term memory as graph and retrieve for responses
- N8n workflow using ZEP: link
- View graph on ZEP:Link

KG Application

From legal documents to knowledge graph:link

KG Terminology

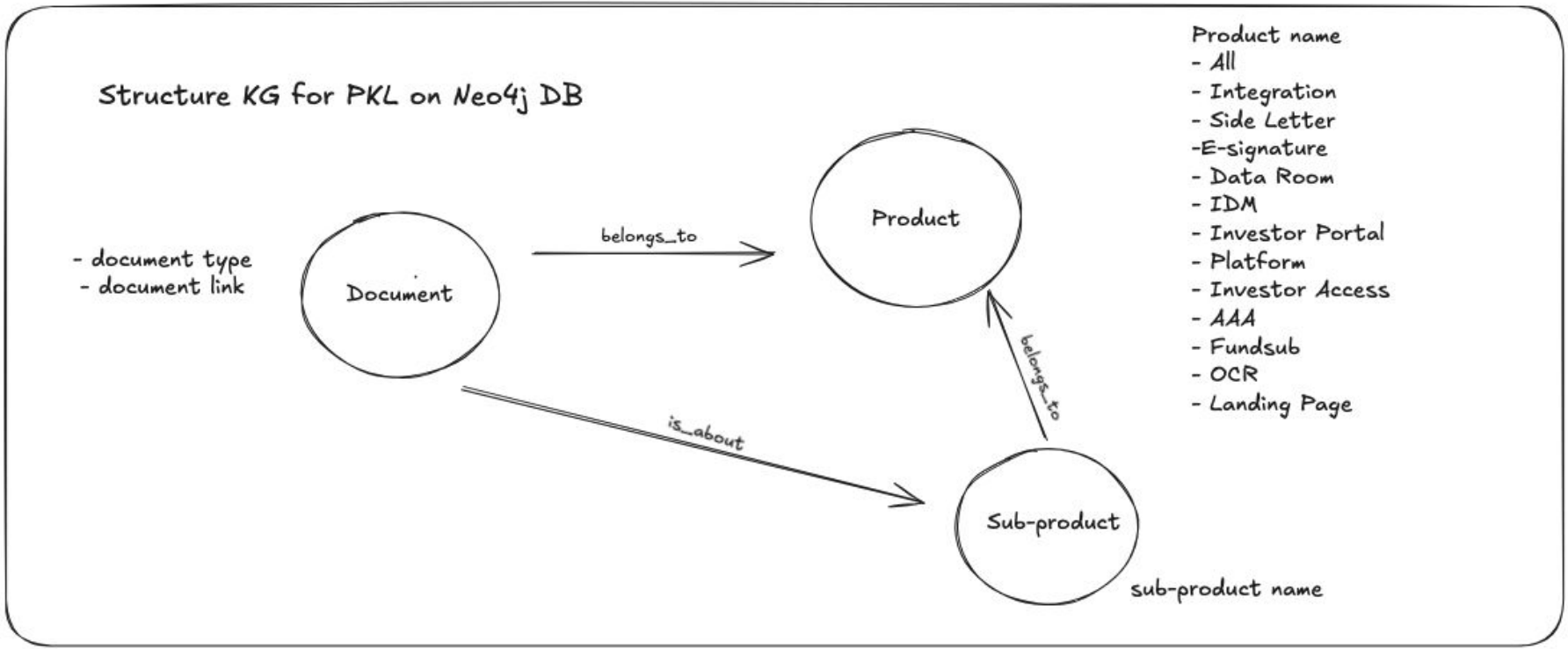
For reference

Term	Simple Definition
Entity	A thing or concept, e.g. a person, company, location or product
Node	A dot in the graph - usually represents an entity
Edge	A line in the graph - represents the relationship between two entities
Triple	A statement in the format: Subject -> Predicate -> Object
Subject	The entity doing or being described
Predicate	The relationship or property
Object	The target of the relationship

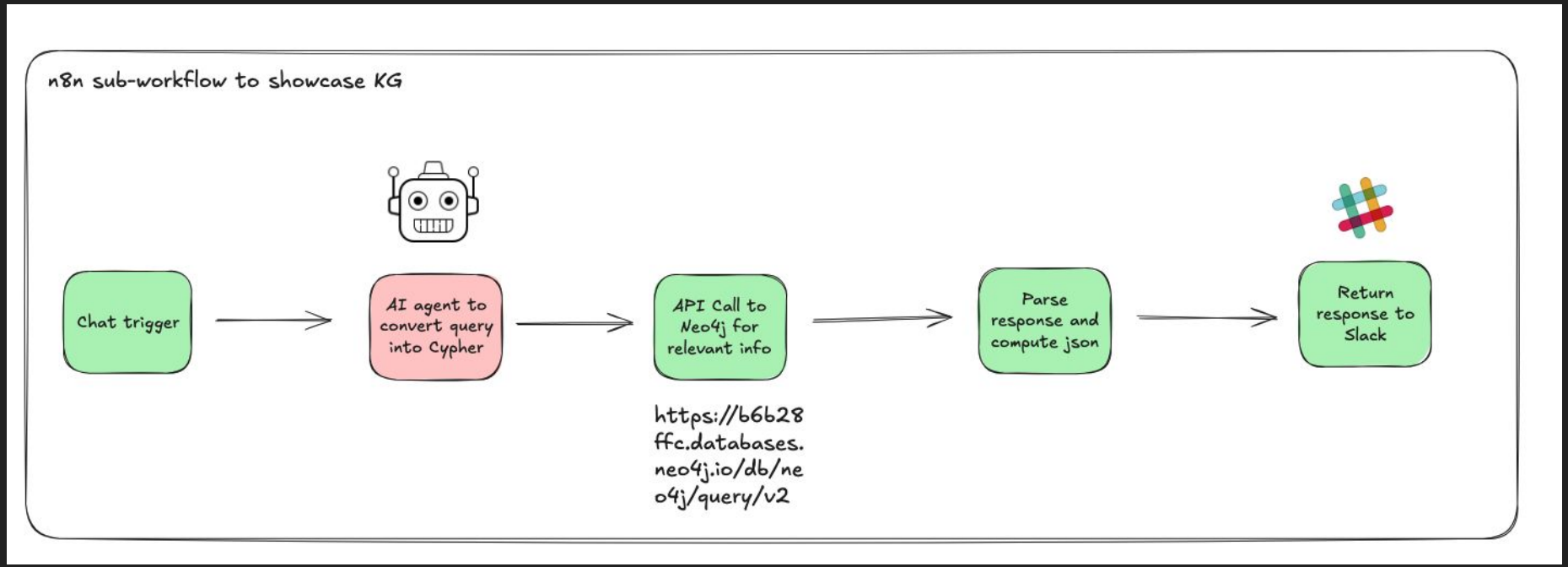
Links & Resources

- Knowledge Graphs for RAG - DeepLearning.AI (link)
- From Local to Global (link)
- GraphRAG (link)
- Demo workflow for KG (link)
- Agentic RAG (link)

Product Knowledge Library



Diagram


n8n sub-workflow to showcase KG
Chat trigger
AI agent to convert query into Cypher
API Call to Neo4j for relevant info
https://b6b28ffc.databases.neo4j.io/db/neo4j/query/v2
Parse response and compute json
Return response to Slack

For reference

Part : Graph solutions

Neo4j, Memgraph, GraphRAG (Microsoft) & LightRAG

The GraphRAG ecosystem

neo4j

LightRAG

GraphRAG