OSA CON 24

Modern LLMs with Graph DB

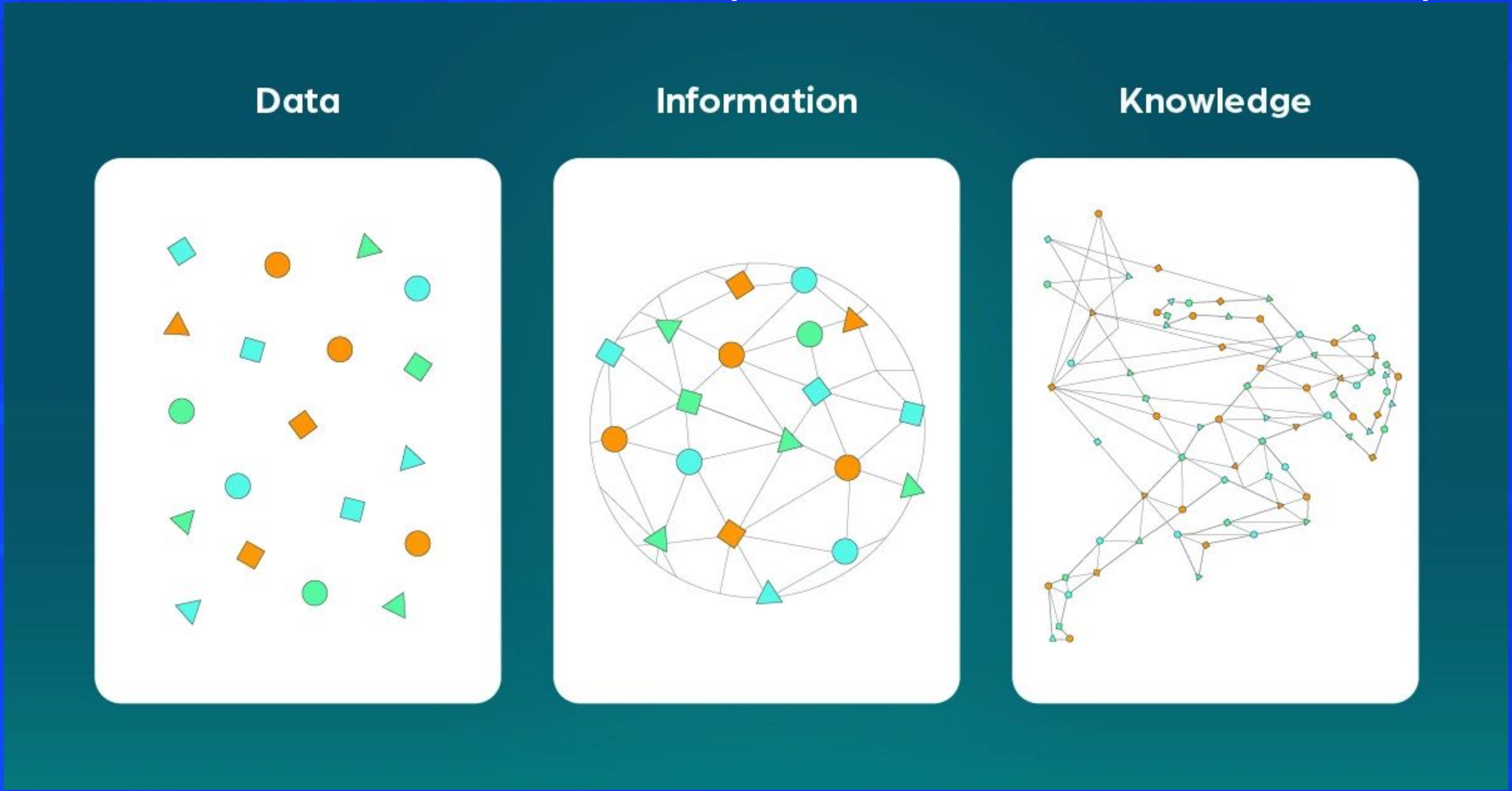
DIPTIMAN RAICHAUDHURI

Staff Developer Advocate - Confluent

https://www.linkedin.com/in/diptimanrc/

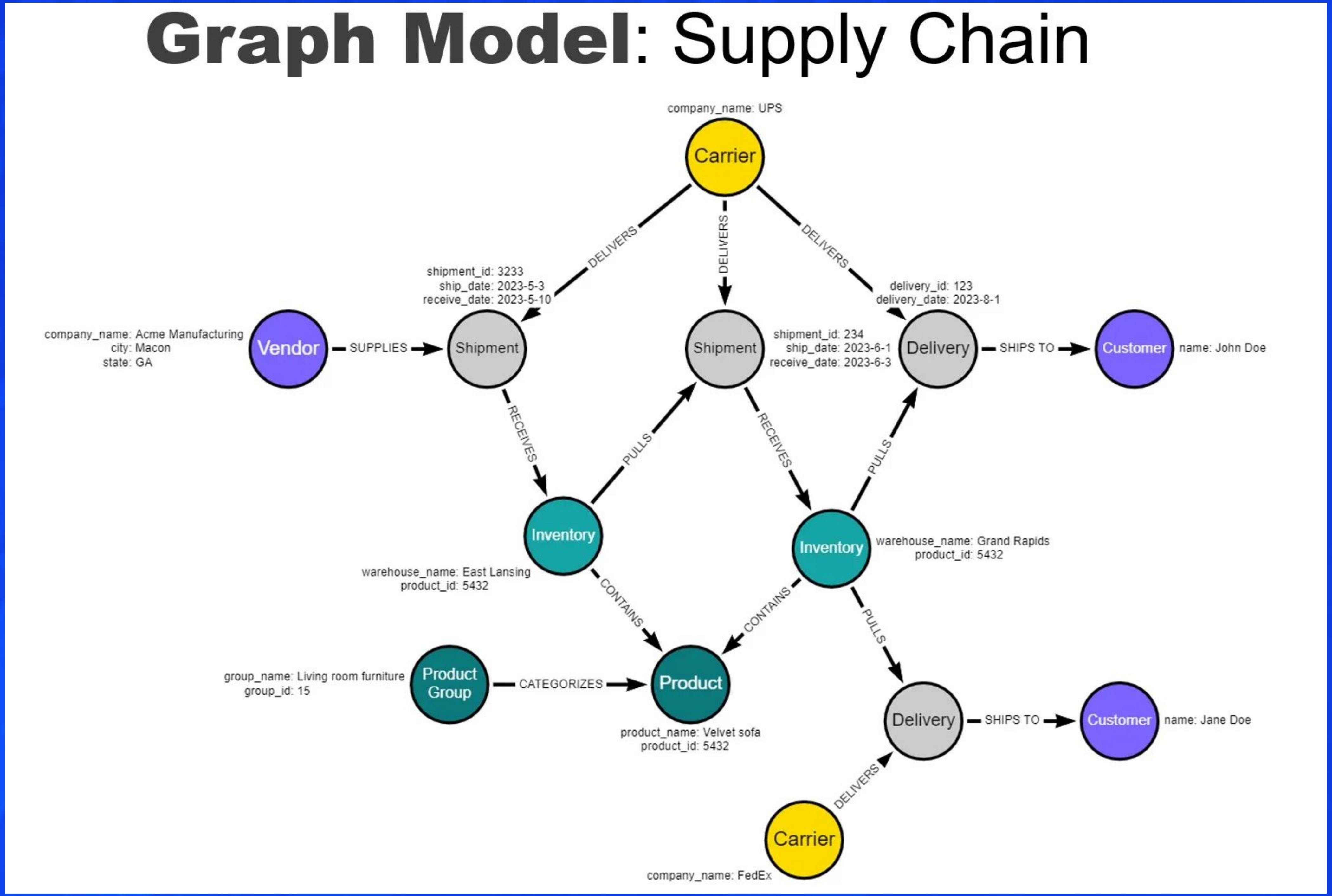
November 19-21, 2024

Quest for Knowledge - All problem is a "search" problem



Source: https://knowmax.ai/blog/data-vs-information-vs-knowledge/

Knowledge Graph



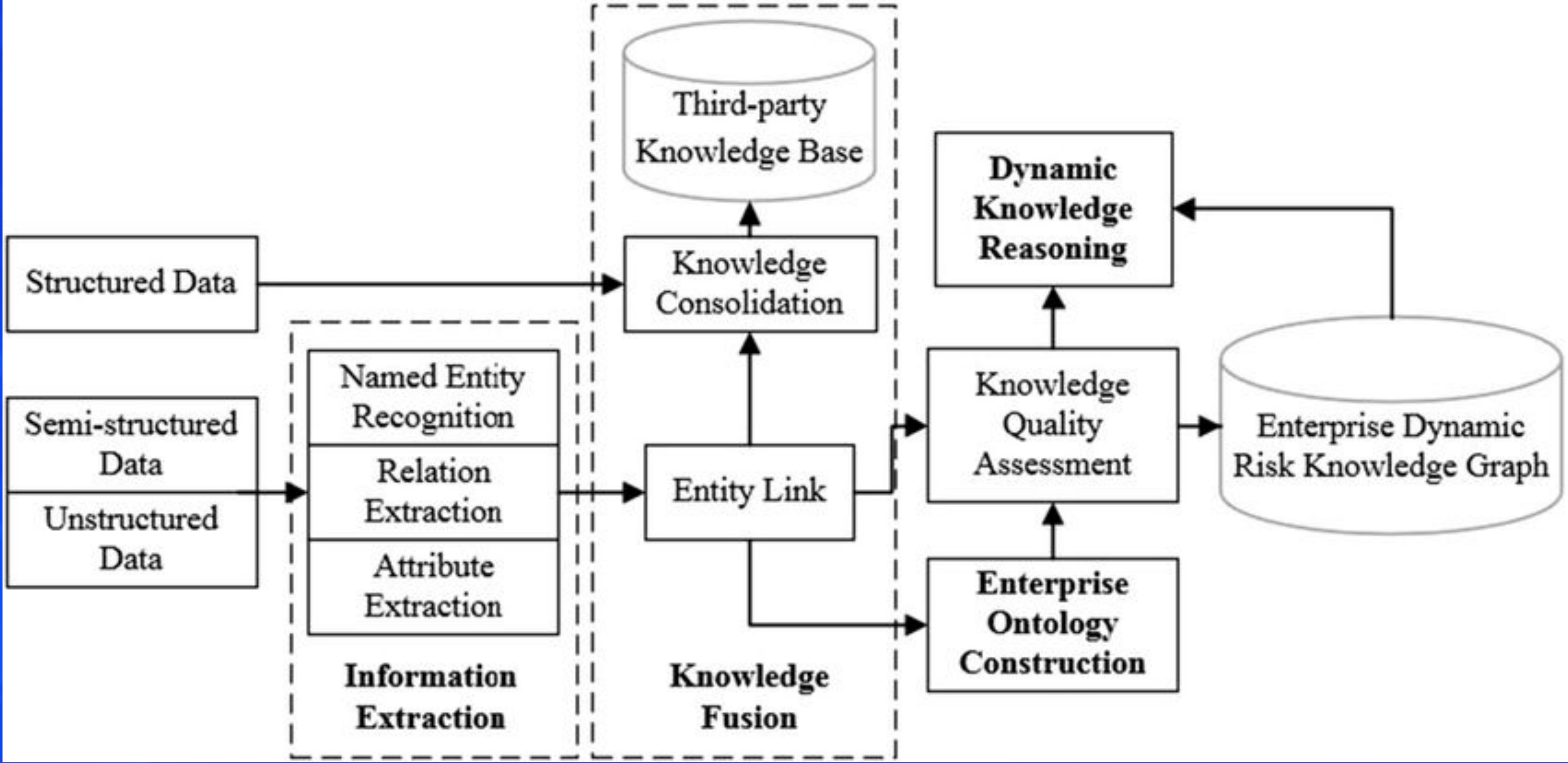
A knowledge graph is an organized representation of real-world entities and their relationships. It is typically stored in a graph database, which natively stores the relationships between data entities.

Reality

Not all structured and unstructured data is a Knowledge Graph ! !

Ontology = Knowledge Graph metadata

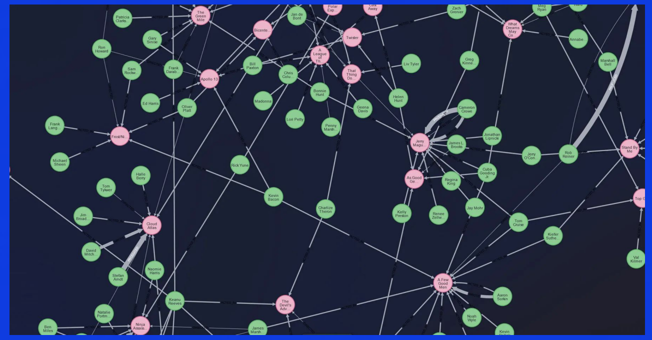
Extracting entities(NER) and relationships based on Ontology is difficult !

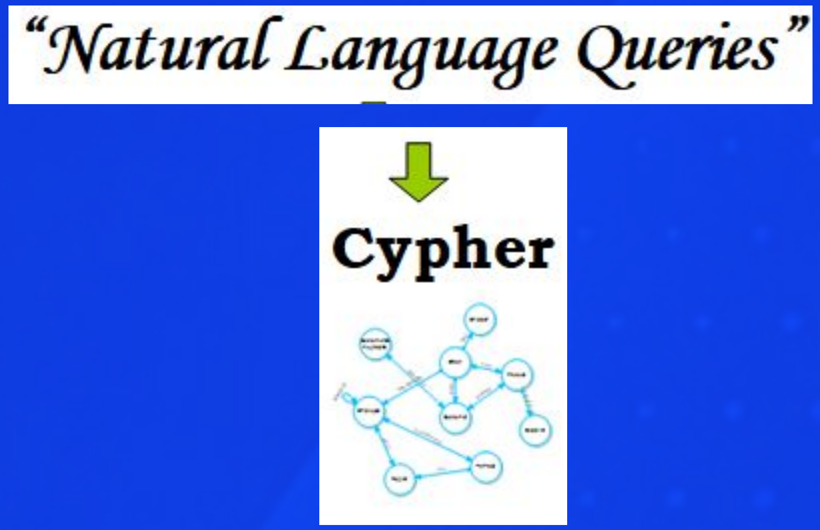


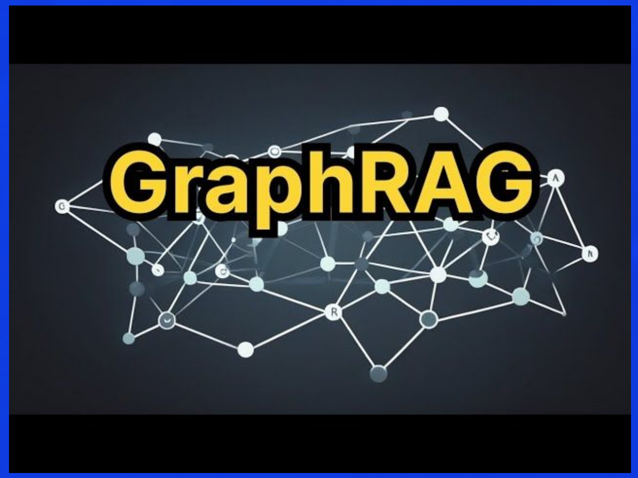
It requires a lot of NLP and Enterprise metadata management tasks

Image Source: Yang, Bo & Liao, Yi-ming. (2022). Research on enterprise risk knowledge graph based on multi-source data fusion. Neural Computing and Applications. 34. 10.1007/s00521-021-05985-w.

Enter LLMs -> Creating Knowledge Graphs



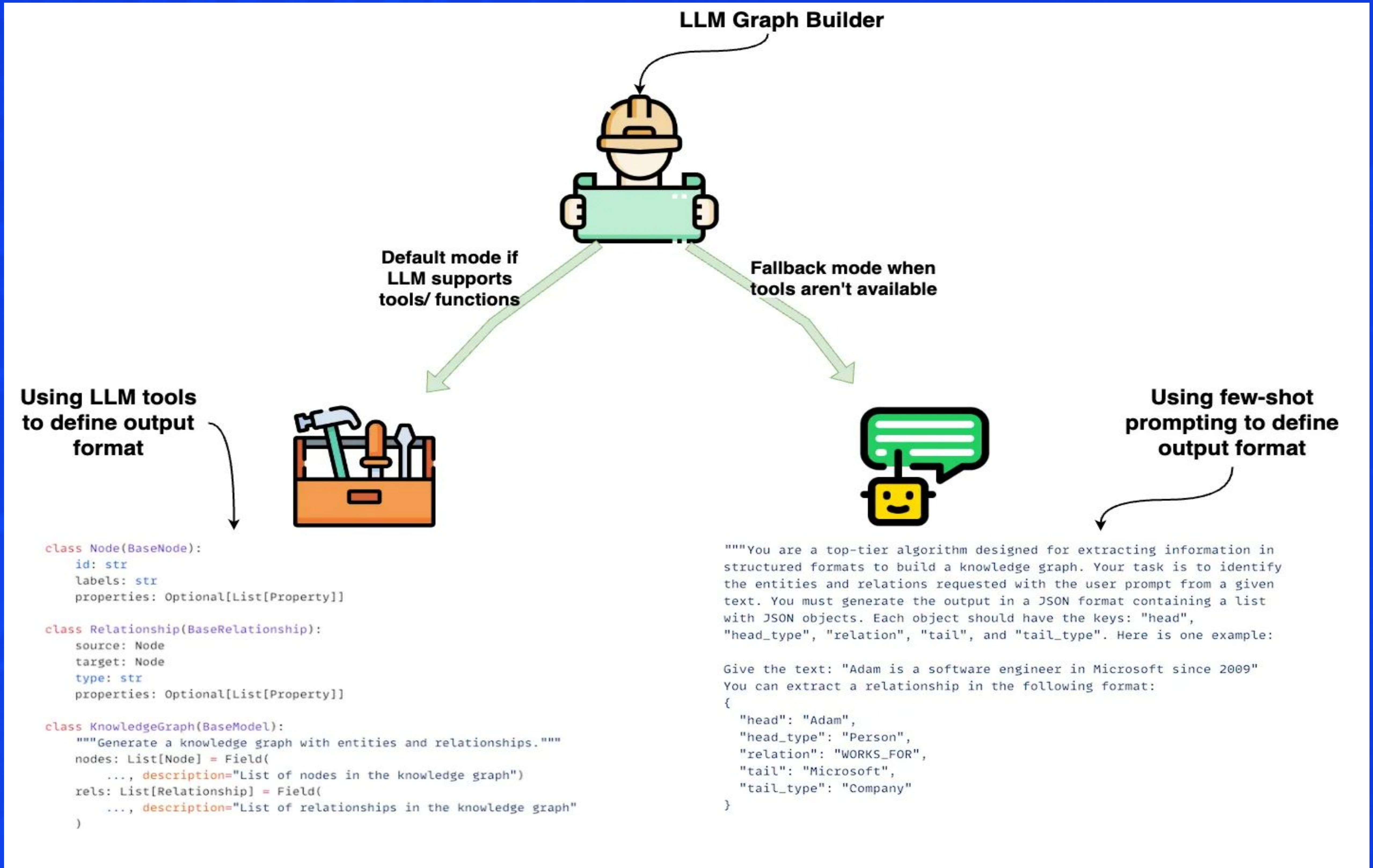




Graph Creation + Graph Query

- Modern LLMs do a decent job in terms of extracting property graphs from unstructured data
- LLMs also convert natural language to cypher queries for querying existing Knowledge Graphs

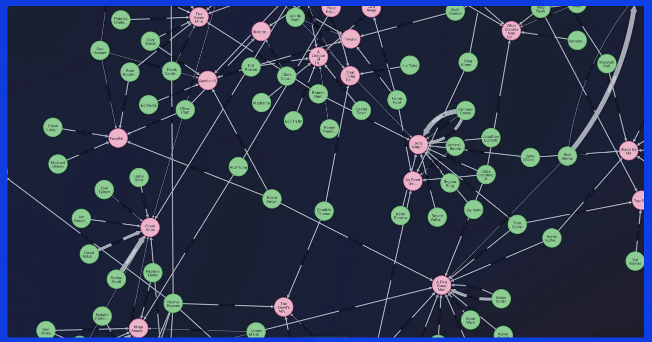
Text -> Knowledge Graph



Source: https://towardsdatascience.com/building-knowledge-graphs-with-llm-graph-transformer-a91045c49b59

Text -> Knowledge Graph

LangChain
LLM Graph Transformer

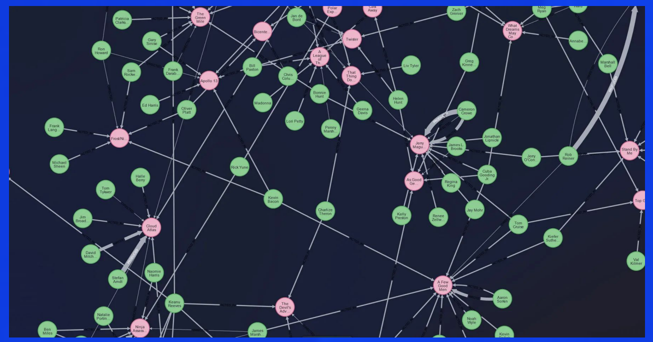


Property Graph

```
llm_transformer = LLMGraphTransformer(llm=llm)
.
.
.
graph_documents = llm_transformer.convert_to_graph_documents(documents)
graph.add_graph_documents(graph_documents_props)
```

Text -> Knowledge Graph

LangChain
LLM Graph Transformer

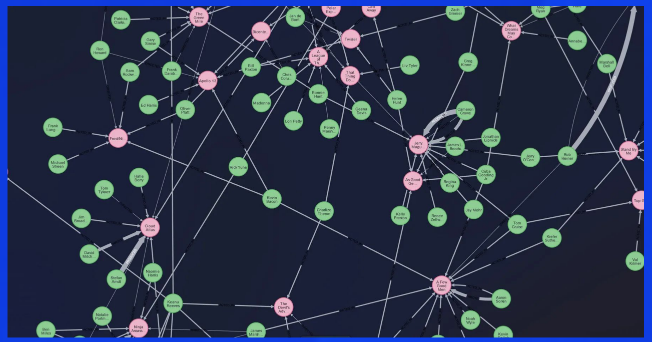


Property Graph

Note that the graph construction process is non-deterministic since we are using LLM.
Therefore, you might get slightly different results on each execution.

Text -> Knowledge Graph (Filtered)

LangChain
LLM Graph Transformer

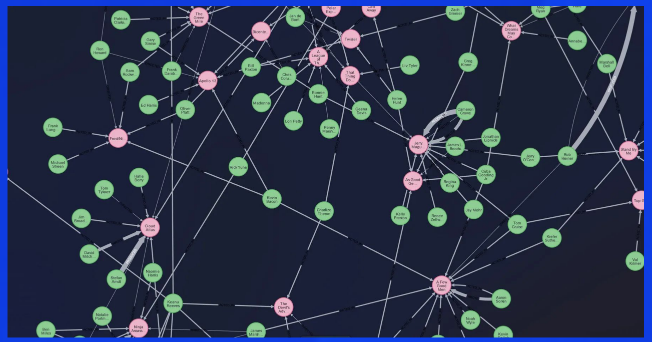


Property Graph
(allowed_nodes,
allowed_relationships)

```
llm_transformer_filtered = LLMGraphTransformer(
    llm=llm,
    allowed_nodes=["Supplier", "OEM", "Organization"],
    allowed_relationships=["HAS_PART", "LOCATED_IN", "CONTRACTED_ON"],
)
.
graph_documents_filtered = llm_transformer_filtered.convert_to_graph_documents(
    documents
graph.add_graph_documents(graph_documents_props)
```

Text -> Knowledge Graph

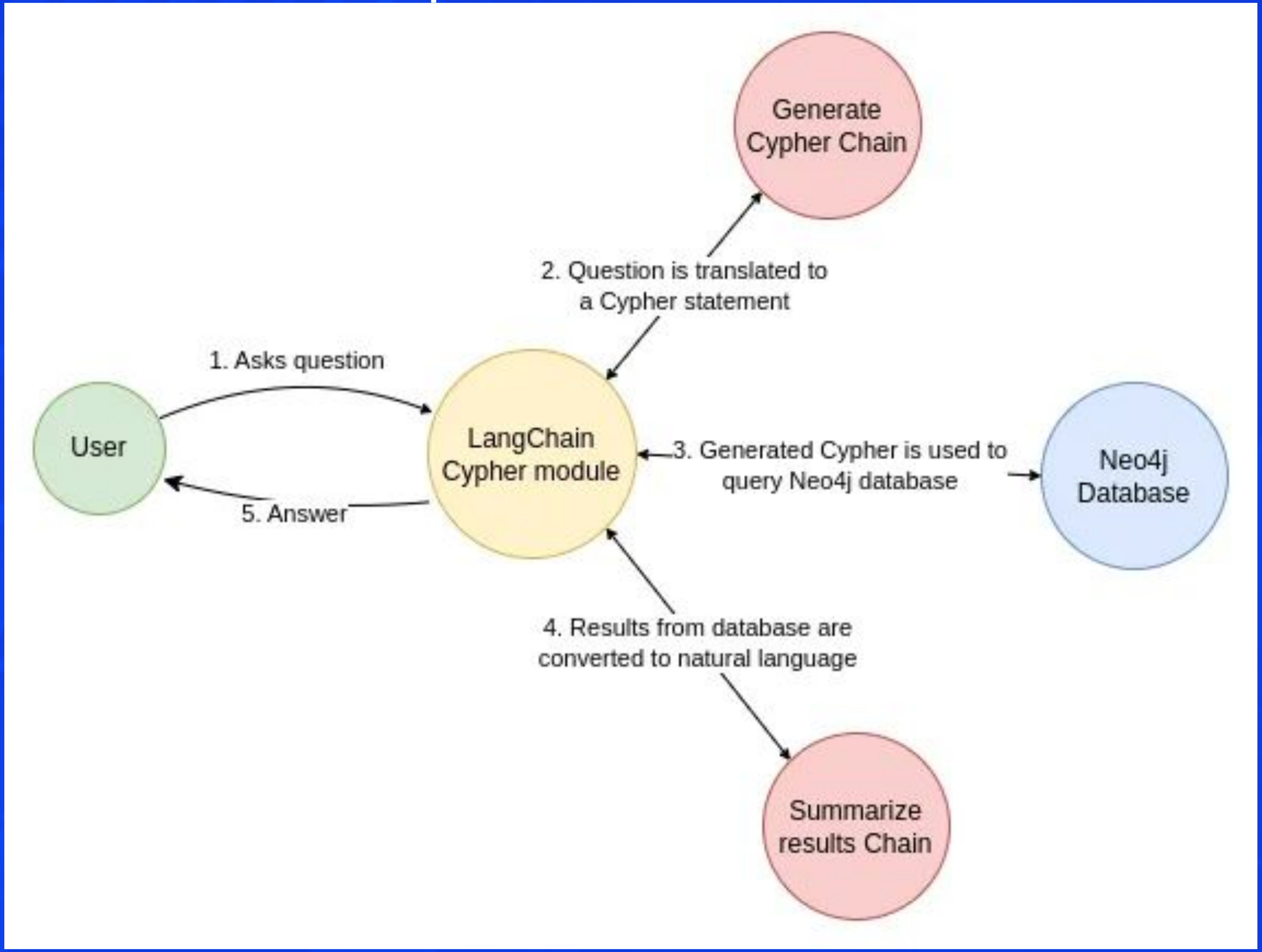
LangChain
LLM Graph Transformer



Property Graph

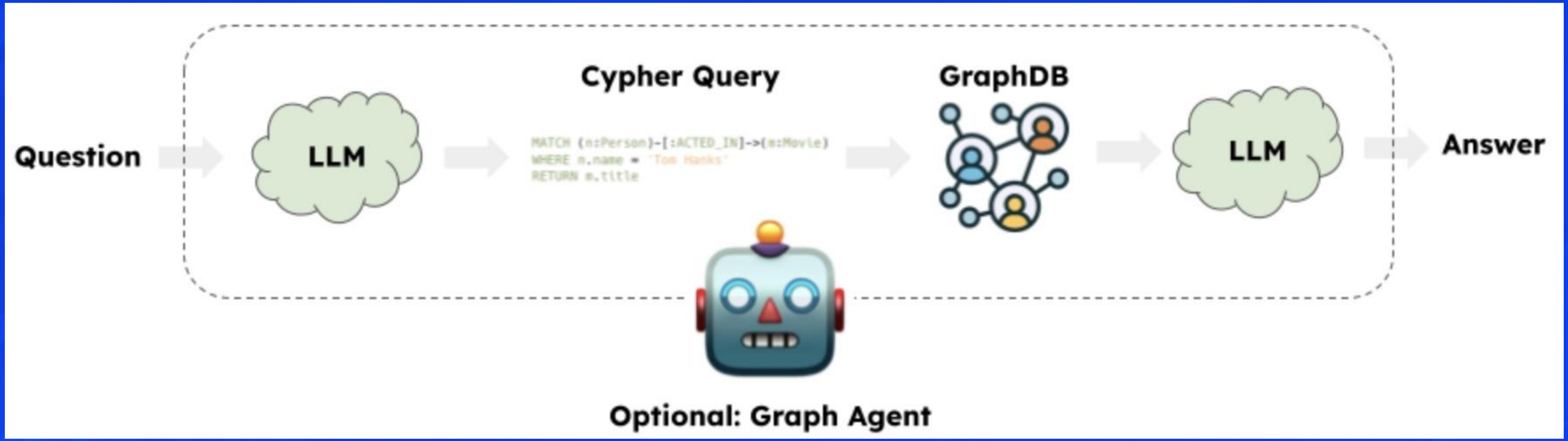
In this case, specific types of nodes and relationships are pre-defined for extraction according to business requirements and to reduce hallucination.

NL2SQL -> text-2-cypher



Source: https://python.langchain.com/v0.1/docs/use_cases/graph/quickstart/

NL2SQL -> text-2-cypher



LLMs use a chain of thought approach, which, at a high level consists of the following steps :

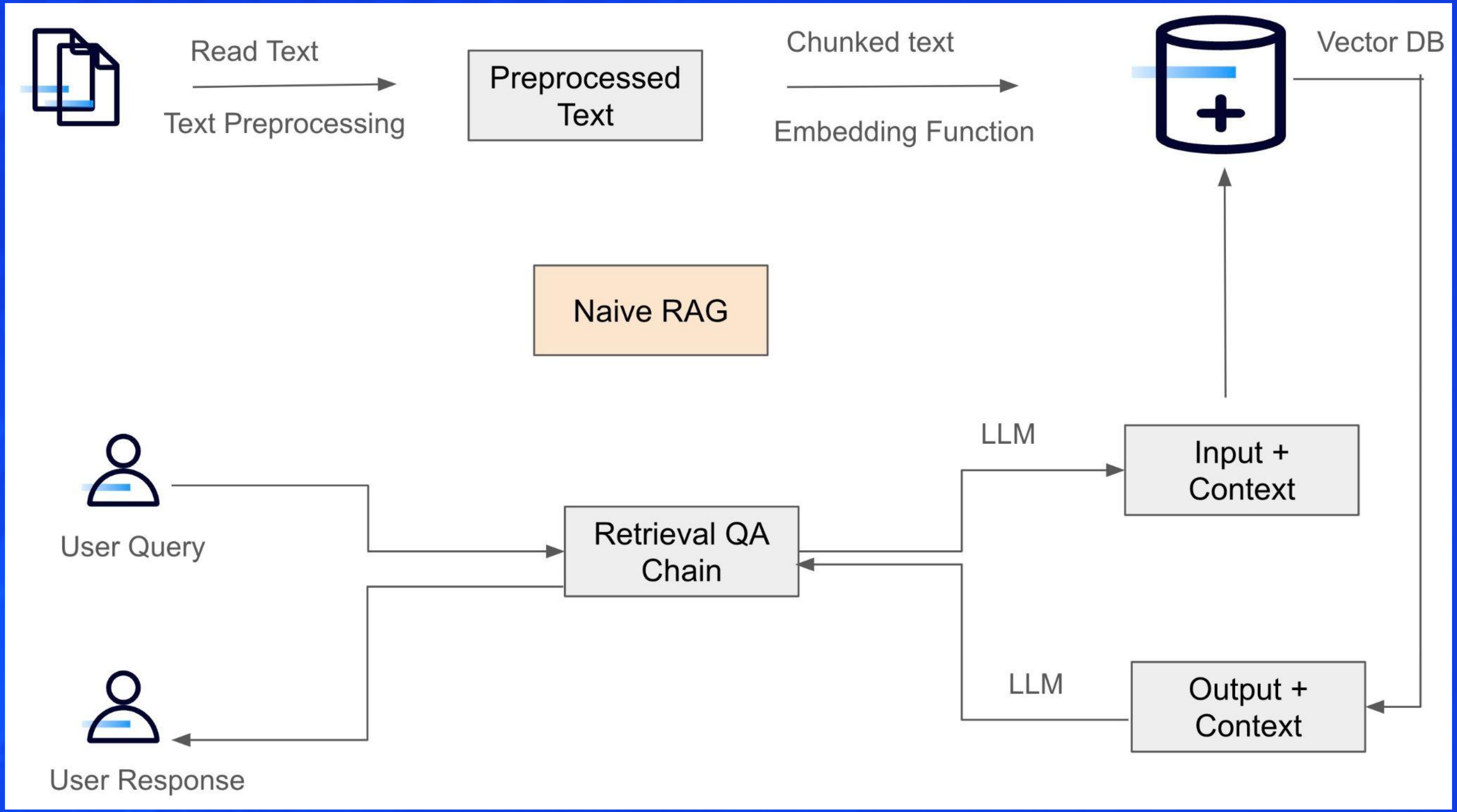
- **Convert question to a graph database query: Model converts user input to a graph database query (e.g. Cypher).**
- **Execute graph database query: Execute the graph database query.**
- **Answer the question: Model responds to user input using the query results.**

NL2SQL -> text-2-cypher

```
from langchain.chains import GraphCypherQAChain
from langchain_community.graphs import Neo4jGraph
from langchain_openai import ChatOpenAI

graph = Neo4jGraph(url=os.environ['NEO4J_URL'],
username=os.environ['NEO4J_USERNAME'],
password=os.environ['NEO4J_PASSWORD'])

graph.refresh_schema()
chain = GraphCypherQAChain.from_llm(
    ChatOpenAI(temperature=0), graph=graph,
verbose=True
)

chain.run("How many employees are there in total?")
```

```
> Entering new GraphCypherQAChain chain...
Generated Cypher:
MATCH (e:Employee)
RETURN COUNT(e) as totalEmployees;
Full Context:
[{'totalEmployees': 9}]

> Finished chain.

'There are 9 employees in total.'
```

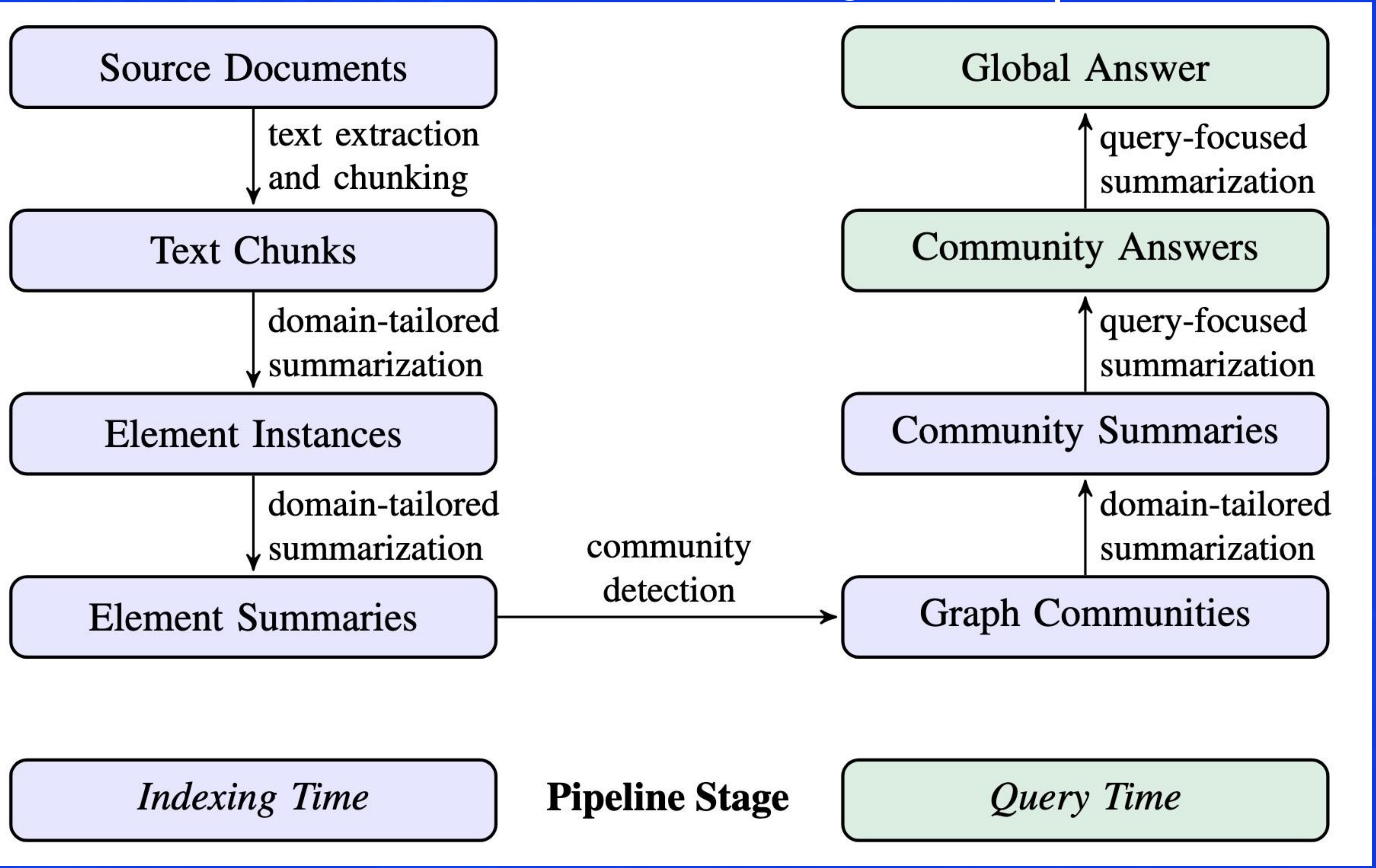
RAG - Retrieval Augmented Generation



Issues with Naive RAG:

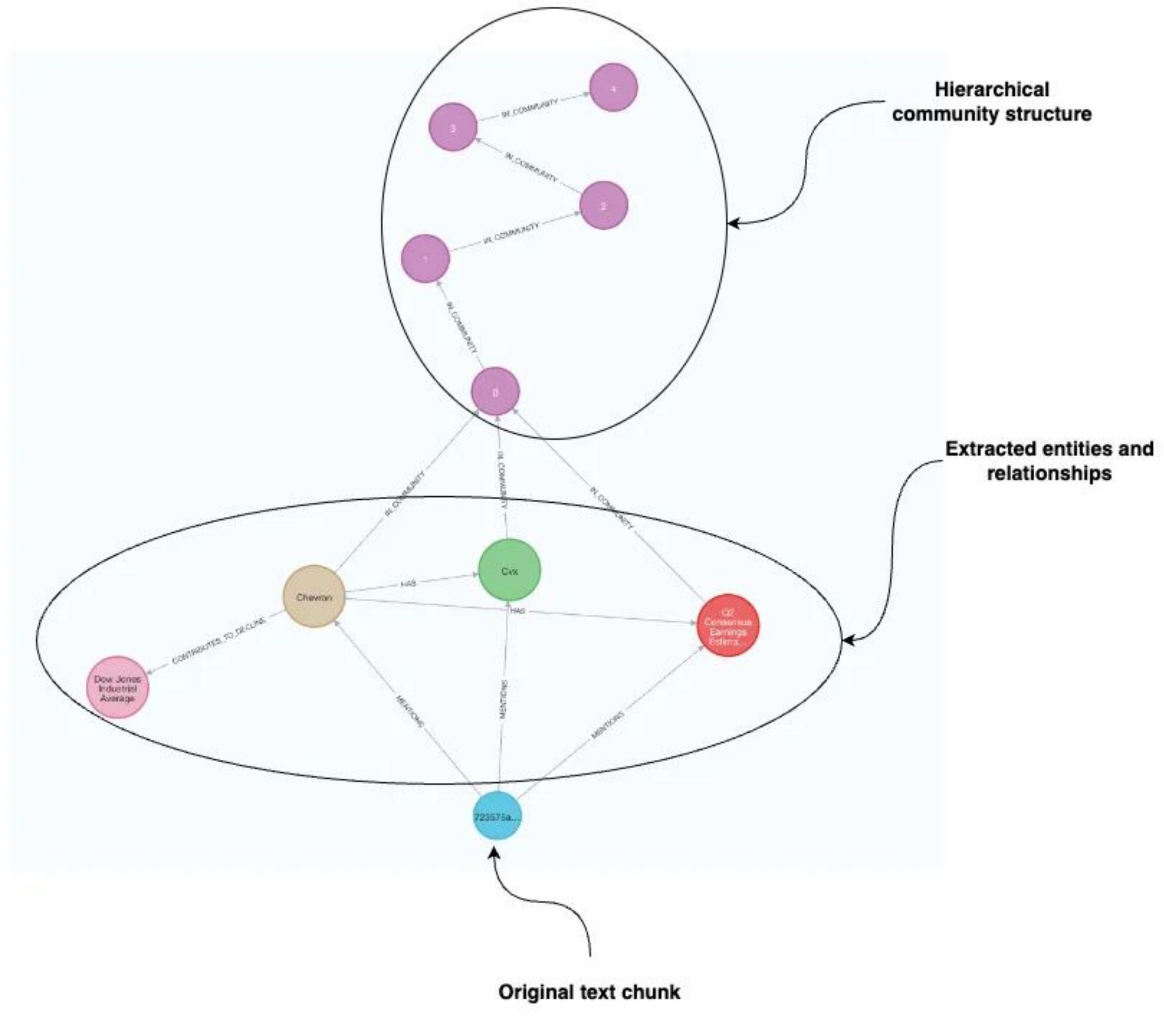
1. Too much focus on similar phrases misses context understanding.
2. Filler text also influences similarity boundaries.
3. Retrieved information used directly without inferring border context

GraphRAG - RAG with Knowledge Graph



Source: https://arxiv.org/pdf/2404.16130

GraphRAG - Advantage



- Knowledge Graph extracted by LLMs to capture semantic structure
- Community detection enriches the structure by introducing densely connected nodes at different levels of granularity
- Delivers much better response compared to summarizing of full text by providing responses that are better connected to the original data

GraphCypherQAChain - Hybrid Search

```
chain = GraphCypherQAChain.from_llm(
    graph=graph, llm=llm, verbose=True, validate_cypher=True
)
response = chain.invoke({"query": "What was the cast of the Casino?"})
response
```

```
[1m> Entering new GraphCypherQAChain chain...[0m
Generated Cypher:
[32;1m[1;3mMATCH (:Movie {title: "Casino"})<-[:ACTED_IN]-(actor:Person)
RETURN actor.name[0m
Full Context:
[32;1m[1;3m[{'actor.name': 'Joe Pesci'}, {'actor.name': 'Robert De Niro'}, {'actor.name': 'Sharon
Stone'}, {'actor.name': 'James Woods'}][0m

[1m> Finished chain.[0m
```

Thank You + Q&A

Reach Me At :

- https://www.linkedin.com/in/diptimanrc/

My Blog

- https://diptimanrc.medium.com/