The GenAI Stack

Andreas Kollegger of Neo4j, a database company

GenerativeAI

GenerativeAI is a Parrot

GenerativeAI is a Sock Puppet

GenerativeAI
is Alien Technology

GenerativeAI

- Learns random sentences from random people
- Talks like a person but doesn't really understand what it's saying
- Occasionally speaks absolute non sense
- Sensitive to question phrasing
- Answers reflect the person asking
- Can't explain or verify answers
- Limited to public "knowledge"

GenerativeAI

- Learns random sentences from random people
- Talks like a person but doesn't really understand what it's saying
- Occasionally speaks absolute non sense
- Sensitive to question phrasing
- Answers reflect the person asking
- Can't explain or verify answers
- Limited to public "knowledge"



How do we integrate with the alien technology?

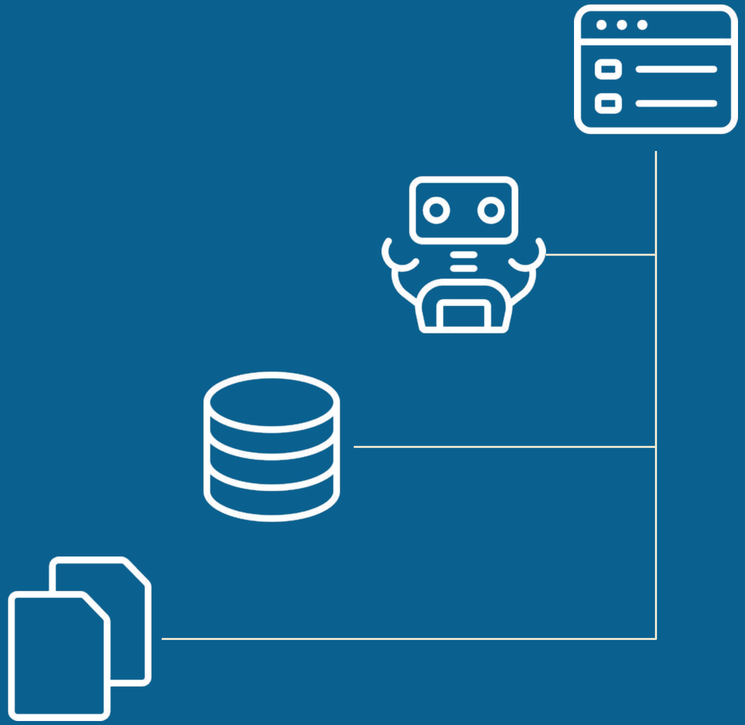
Generative AI is a new layer in the Stack

Application

GenAI

Database

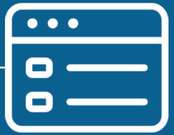
Files



GenAI Stack with Neo4j

Application

LangChain



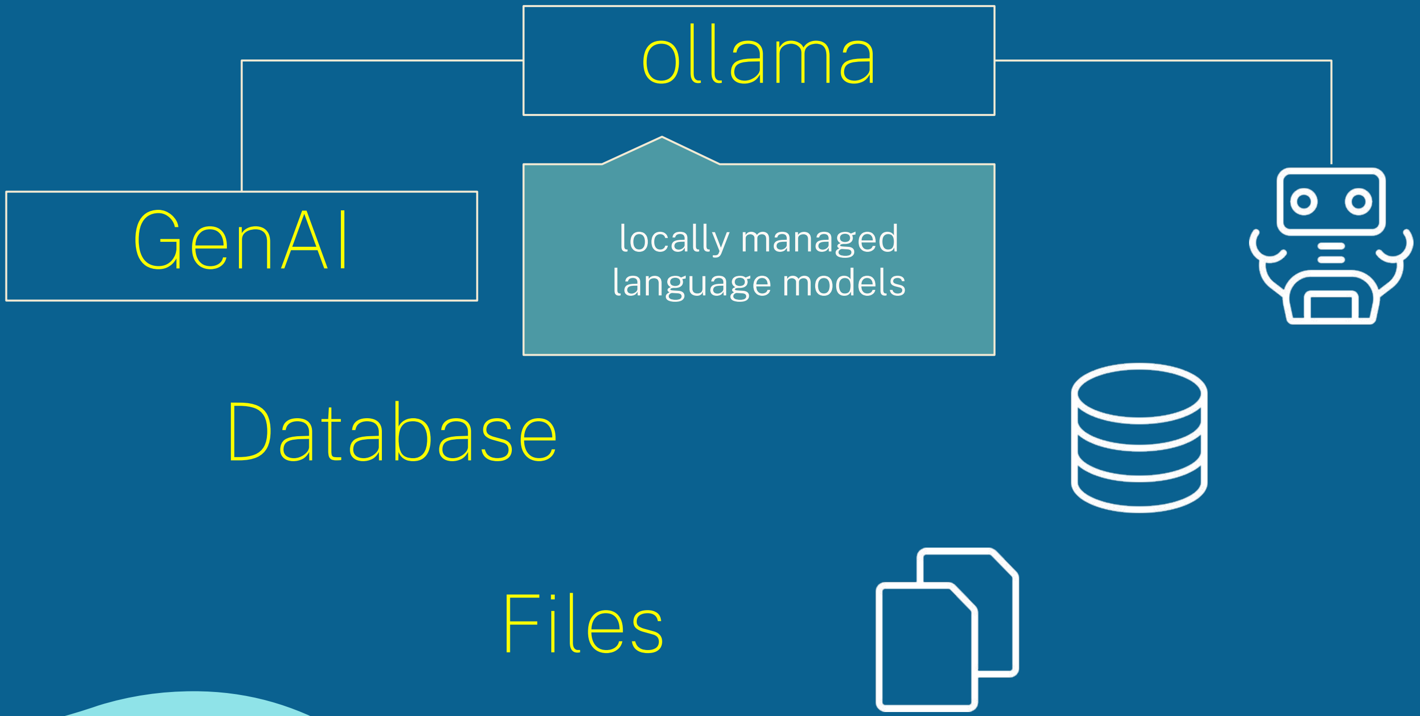
an "orchestration framework" for integrating with LLMs

GenAI

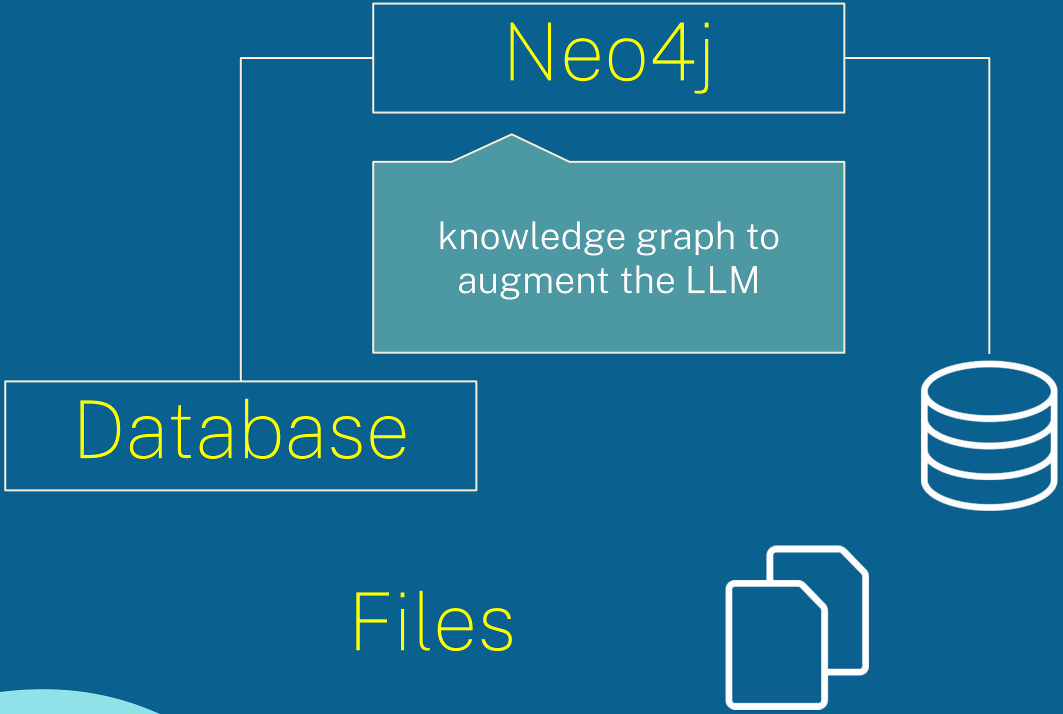
Database

Files

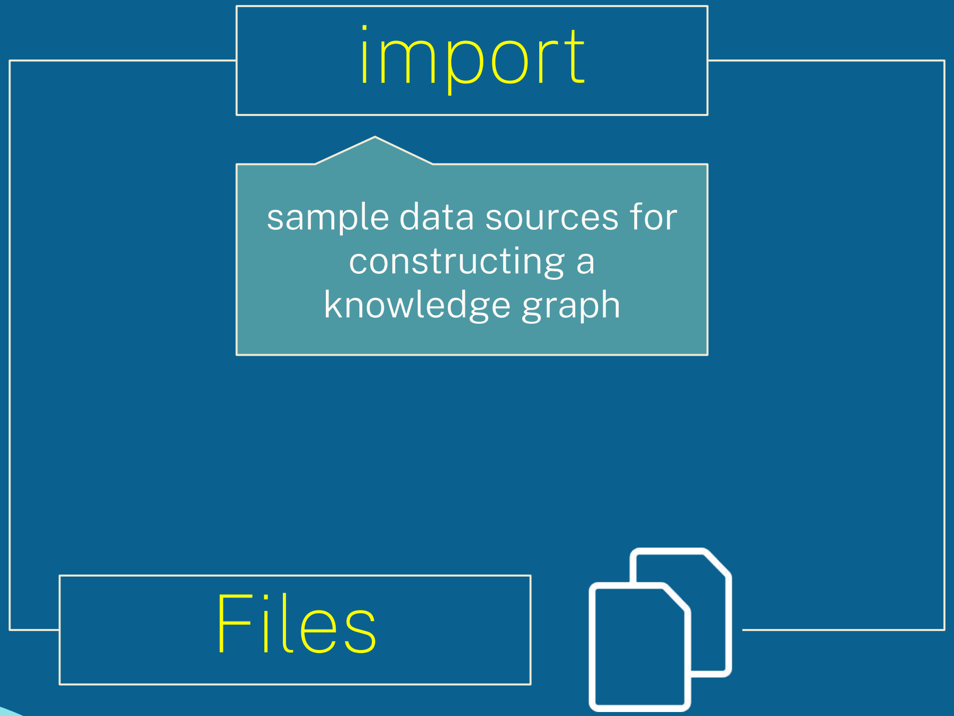
GenAI Stack with Neo4j



GenAI Stack with Neo4j



GenAI Stack with Neo4j



GenAI Stack with Neo4j

LangChain

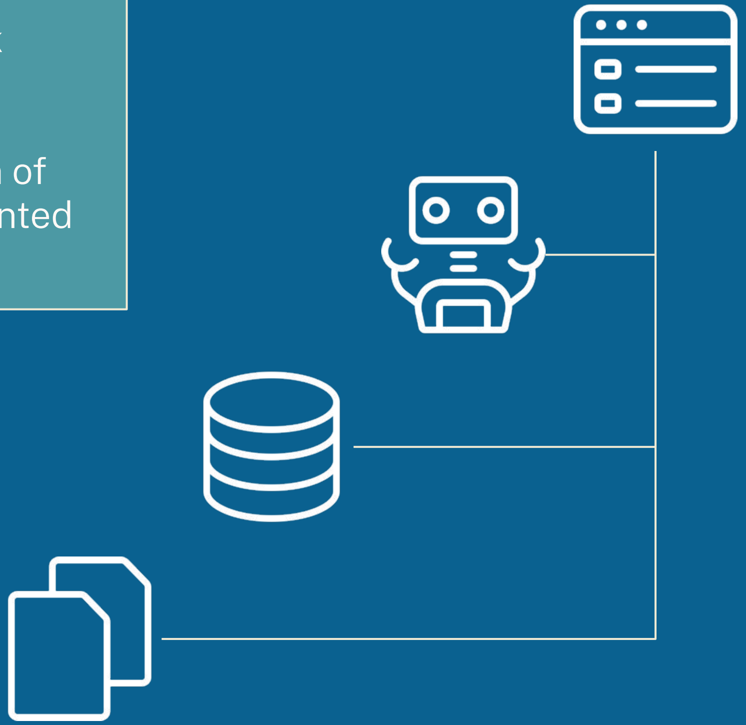
Ollama

Neo4j

Files

GenAI Stack

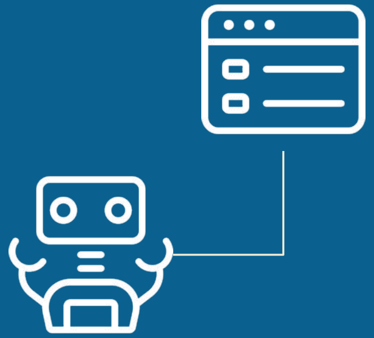
A reference implementation of Retrieval Augmented Generation



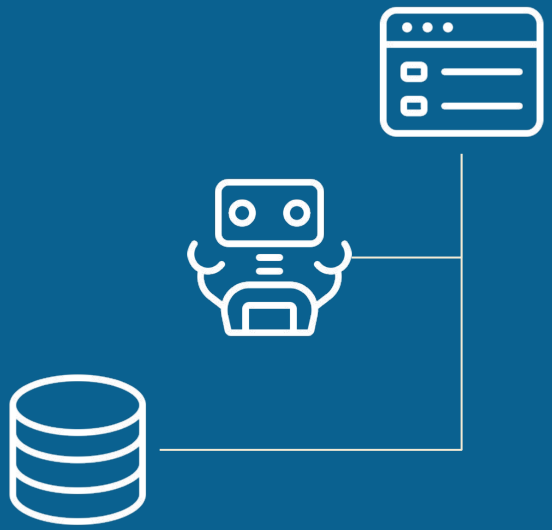
Retrieval Augmented Generation (RAG)

A Generative AI app uses an LLM to provide answers

(aka ChatGPT)



RAG augments the LLM by making requests to a database



Three Data Access Scenarios for RAG

1) Pure Text

unstructured data

typically PDFs or other text documents

2) Mixed Text + Data

structured data with long-form text

database used for CMS

3) Pure Data

structured data with short text values

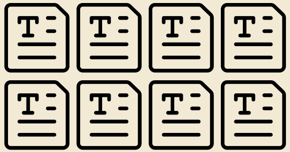
general database

Three Data Access Scenarios for RAG

1) Pure Text

unstructured data

typically PDFs or other text documents



2) Mixed Text + Data

structured data with long-form text

database used for CMS

3) Pure Data

structured data with short text values

general database

Three Data Access Scenarios for RAG

1) Pure Text

unstructured data

typically PDFs or other text documents

2) Mixed Text + Data

structured data with long-form text

database used for CMS

3) Pure Data

structured data with short text values

general database

Three Data Access Scenarios for RAG

1) Pure Text

unstructured data

typically PDFs or other text documents

2) Mixed Text + Data

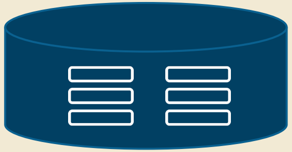
structured data with long-form text

database used for CMS

3) Pure Data

structured data with short text values

general database

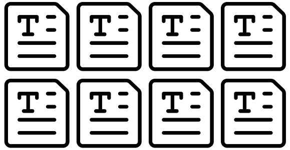


Three Data Access Scenarios for RAG

1) Pure Text

unstructured data

typically PDFs or other text documents



2) Mixed Text + Data

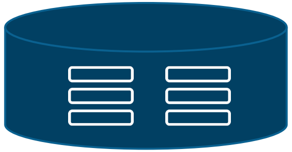
structured data with long-form text

database used for CMS

3) Pure Data ?

structured data with short text values

general database

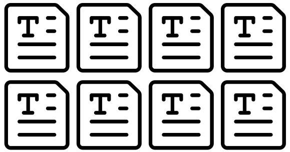


Three Data Access Scenarios for RAG

1) Pure Text

unstructured data

typically PDFs or other text documents



2) Mixed Text + Data

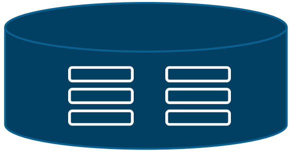
structured data with long-form text

database used for CMS

3) Pure Data

structured data with short text values

general database

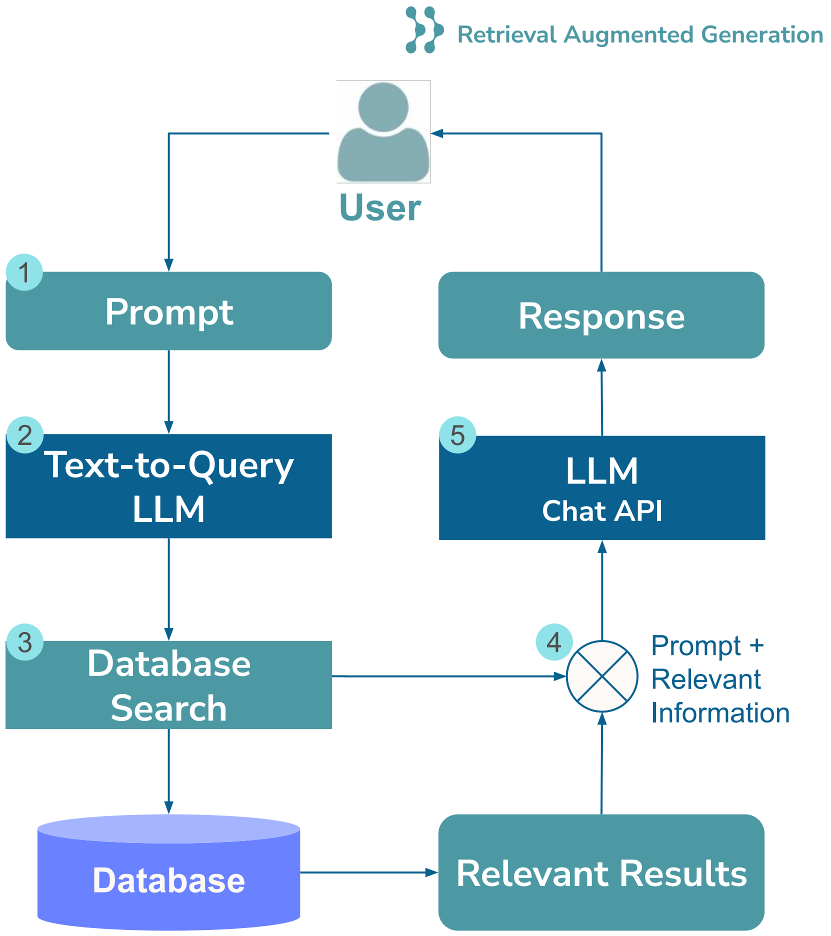


Text is the coupling to the Alien Technology

3) Pure Data Retrieval

Pure Data Retrieval

1. accept prompt from user
2. pass prompt to a fine-tuned code generation model
3. run query against database
4. combine user prompt with query results
5. generate natural language results with LLM



neo4j

3) Pure Data Retrieval Challenges

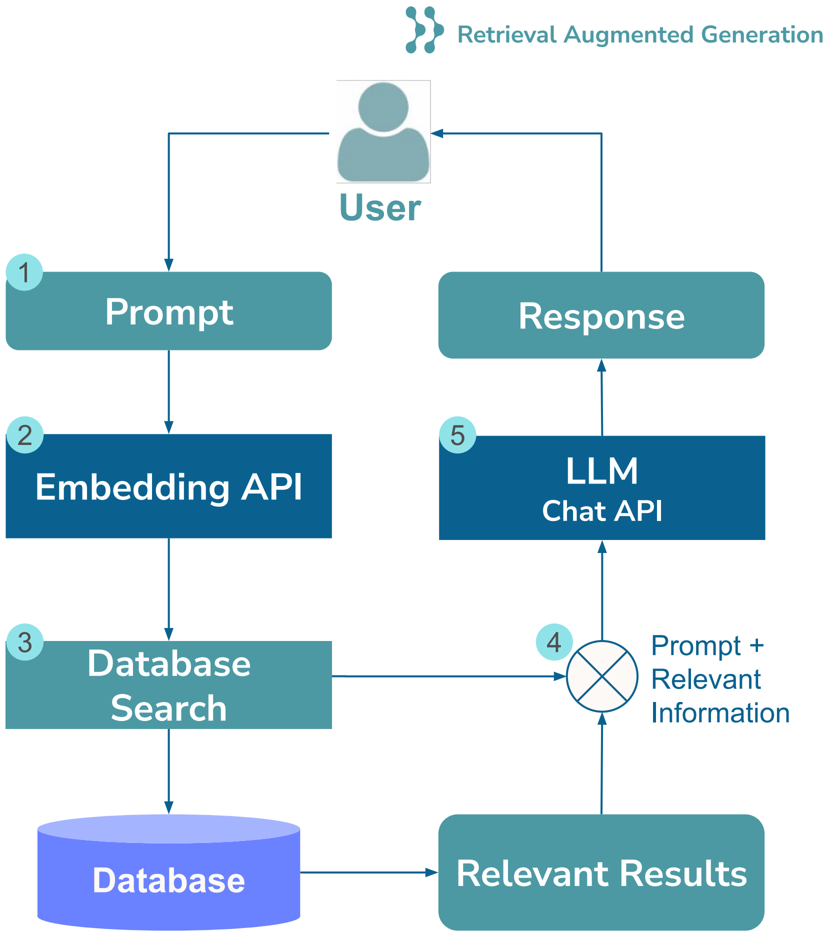
- Getting it to work at all – generating syntactically correct queries
- Getting it to do the right thing – producing meaningful results
- Avoiding accidents – mistaken deletion
- Preventing malicious intent – SQL injection gone wild

An area of active research and development.

1) Pure Text Retrieval

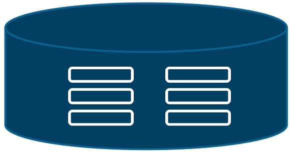
Pure Text Retrieval

1. accept prompt from user
2. **generate an embedding for the prompt**
3. **perform a vector/similarity search on the embedding**
4. combine user prompt with search results
5. generate natural language results with LLM



Pure Text Preparation

1. Chunking	2. Embedding	3. Persistence
Lorem ipsum dolor sit amet, consectetur	[0.2,0.2,0.1,0.7]	
adipiscing elit, sed do eiusmod tempor	[0.3,0.2,0.1,0.5]	
incididunt ut labore et dolore magna aliqua. Ut	[0.4,0.2,0.1,0.7]	
enim ad minim veniam, quis nostrud	[0.5,0.2,0.1,0.7]	
exercitation ullamco laboris nisi ut aliquip	[0.6,0.2,0.1,0.7]	



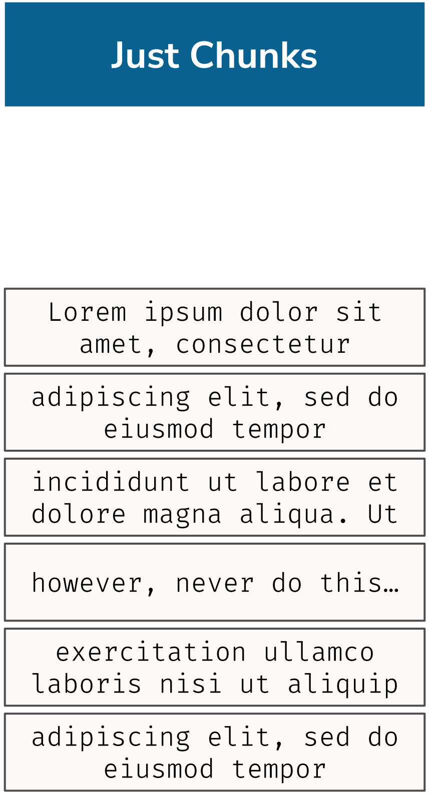
Pure Text, Just Chunks

How?

- pick a chunk method & size
- each chunk is a record
- store chunk with metadata

Challenges:

- what makes a good chunk?
- potential chunk duplication
- how to re-assemble chunk context?



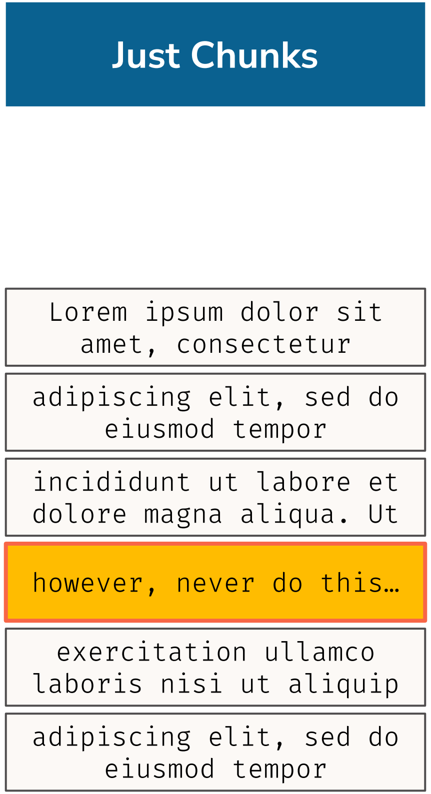
Pure Text, Just Chunks

How?

- pick a chunk method & size
- each chunk is a record
- store chunk with metadata

Challenges:

- what makes a good chunk?
- potential chunk duplication
- how to re-assemble chunk context?



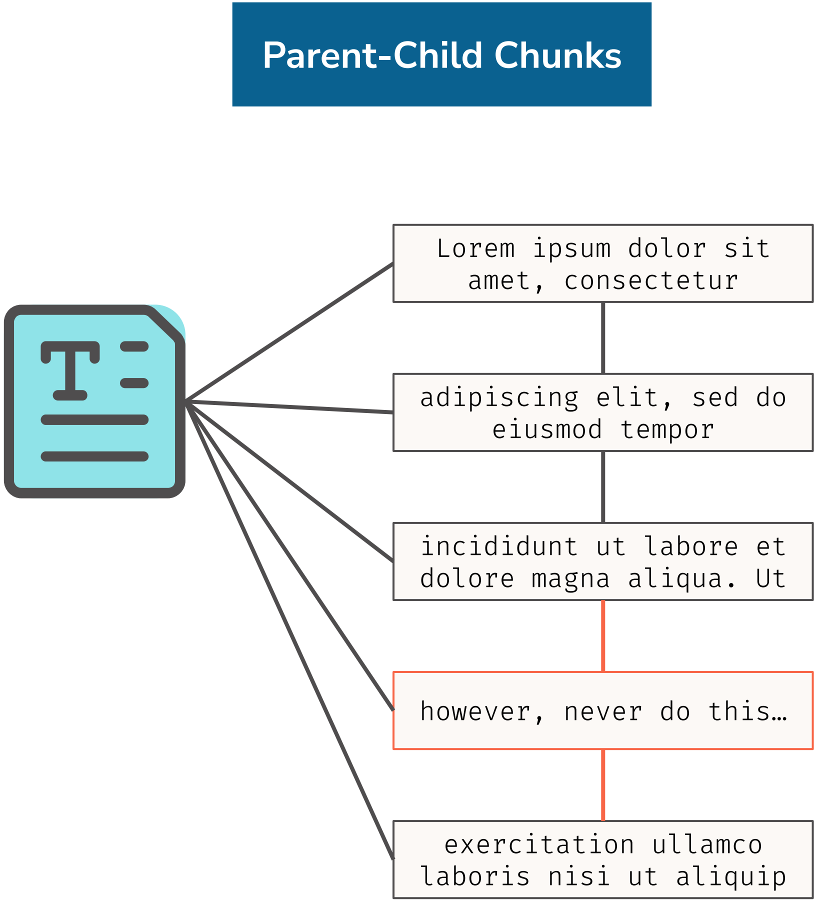
Pure Text, Parent-Child

How?

- connect each chunk to original document
- connect previous/next chunk

Challenges:

- what about cross-document chunks?
- explaining the relevance



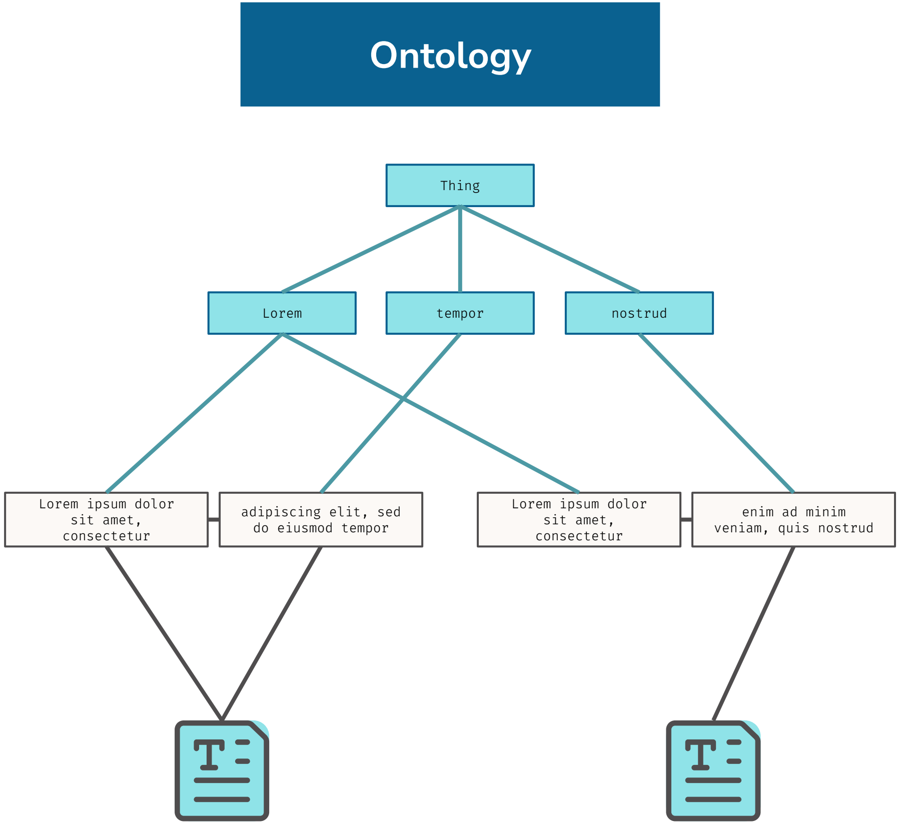
Explicit Similarity

How?

-
- named entity recognition
- metadata extraction
- document cross-linking, page ranking

Challenges:

- does chunk similarity provide the most relevant answer?
- what about the person asking the question?



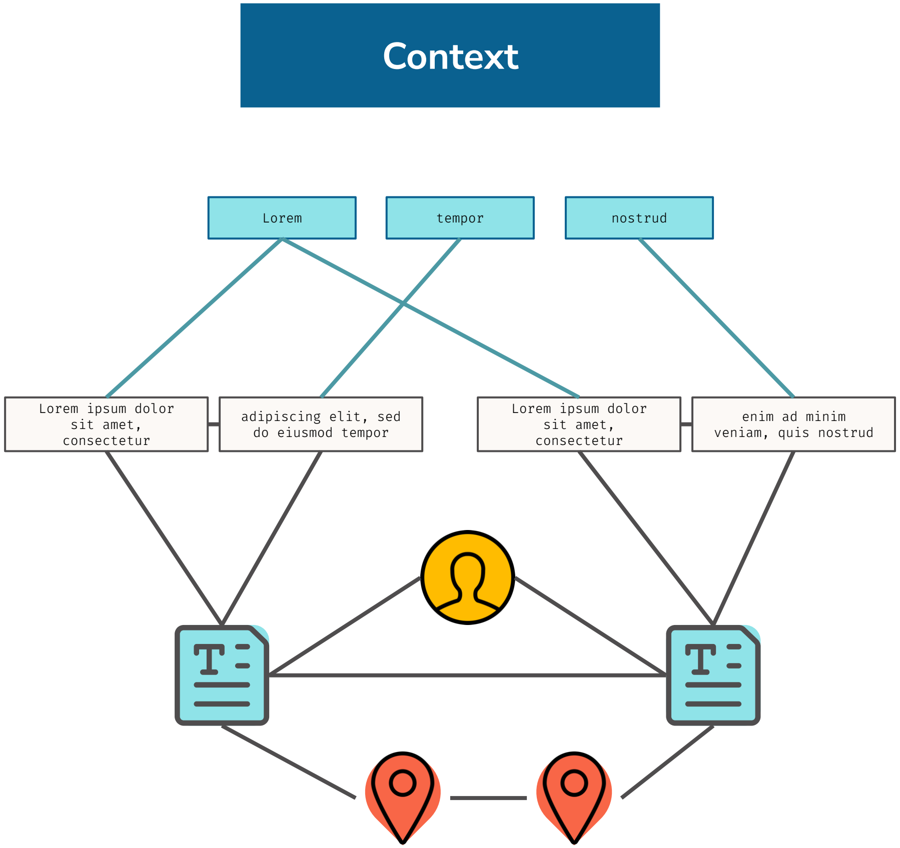
Pure Text, Context

How?

- named entity recognition
- metadata extraction
- document cross-linking, page ranking

Challenges:

- does chunk similarity provide the most relevant answer?
- what about the person asking the question?



Text in, find relevant text, return text

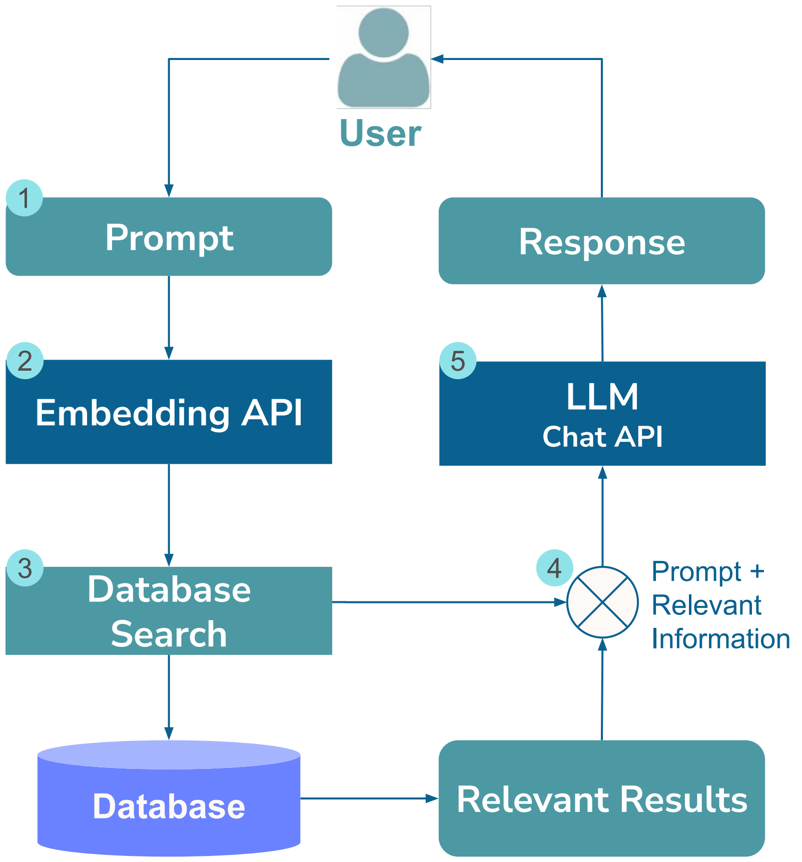
Lorem ipsum

2) Mixed Text & Data Retrieval

Mixed Text & Data Retrieval

1. accept prompt from user
2. generate an embedding for the prompt
3. **perform a data query, anchored on the embedding**
4. combine user prompt with search results
5. generate natural language results with LLM

Retrieval Augmented Generation



neo4j

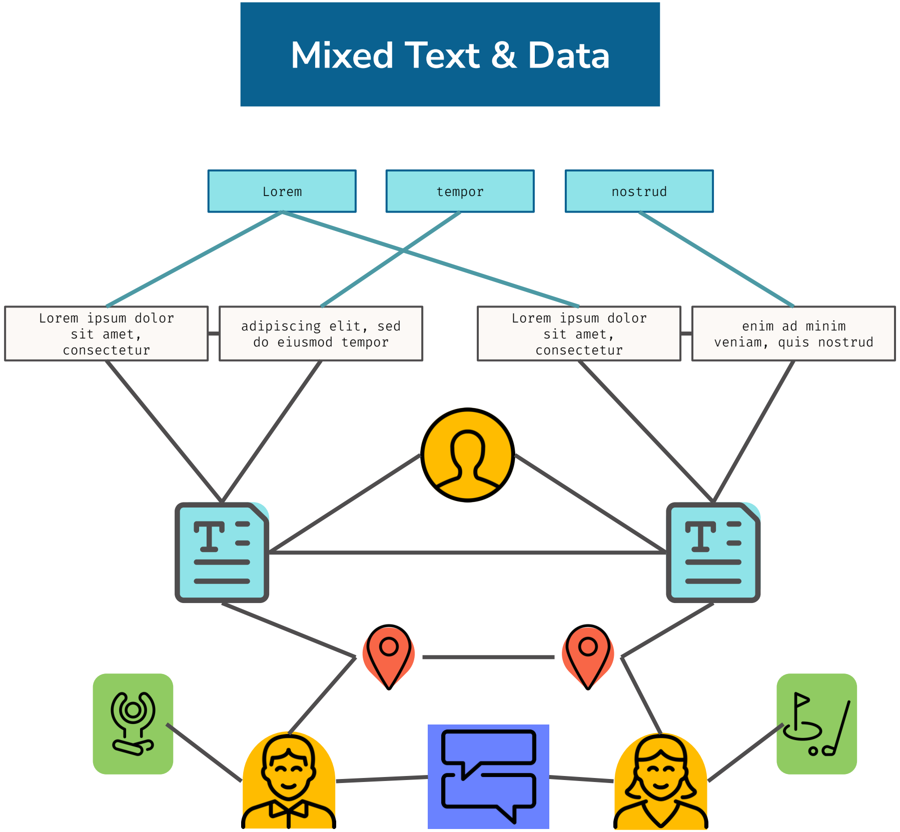
Mixed Text & Data

How?

- extracted information
- application data
- user data
- explicit semantic connections

Challenges:

- what is most relevant,
  to the user?



Text plus data for complete, relevant answers

Lorem ipsum

Knowledge Graphs

A Knowledge Graph is a structured way of representing information, typically using nodes and edges to depict relationships between entities (e.g., people, places, things, concepts).

These entities and their interconnections form a graph-like structure, which can be used to model complex sets of data and the relationships within that data.

– ChatGPT

In knowledge representation and reasoning, a Knowledge Graph is a knowledge base that uses a graph-structured data model or topology to represent and operate on data. Knowledge graphs are often used to store interlinked descriptions of entities – objects, events, situations or abstract concepts – while also encoding the semantics or relationships underlying these entities.[1]

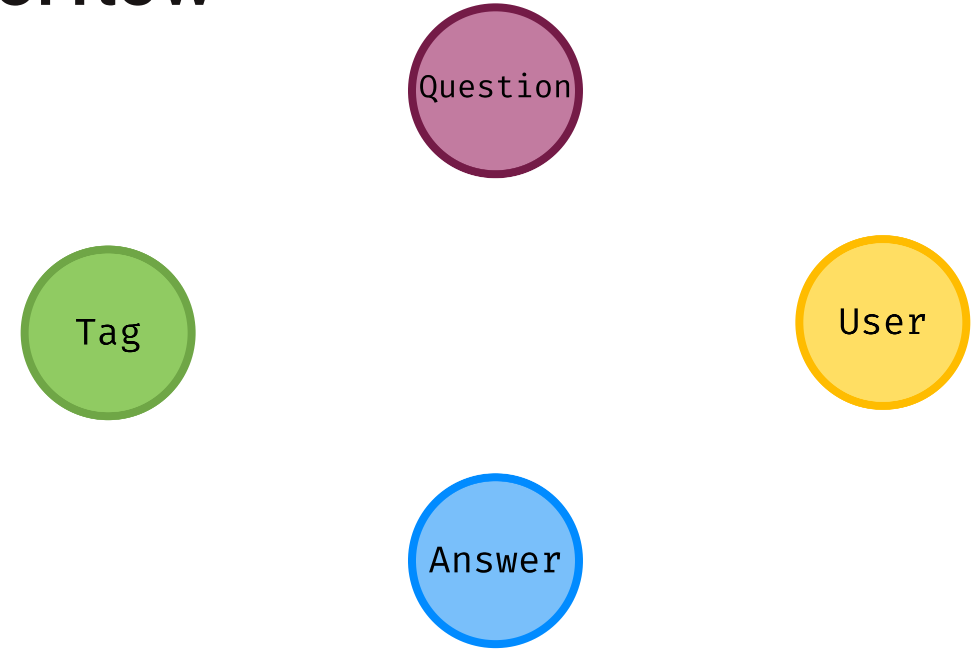
– Wikipedia

A Knowledge Graph is a data structure
where information is stored
in both **objects** and
the **relationships** between objects.

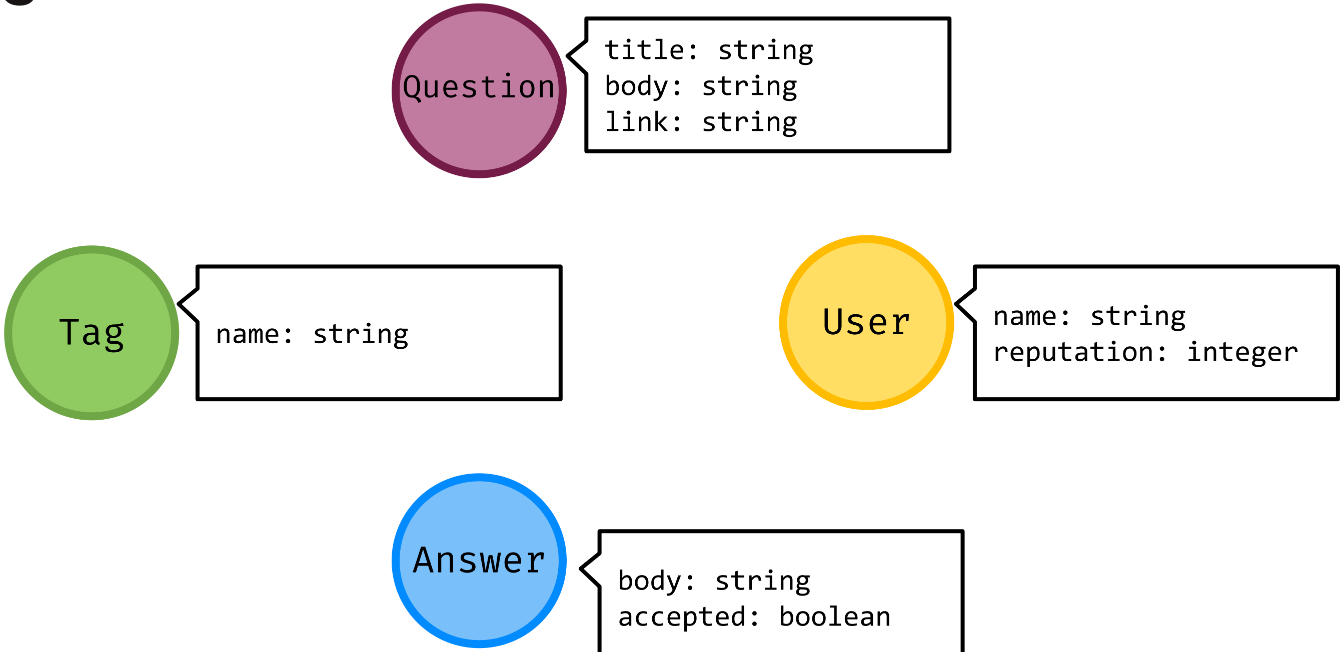
– Andreas Kollegger

For example,
StackOverflow as a Knowledge Graph

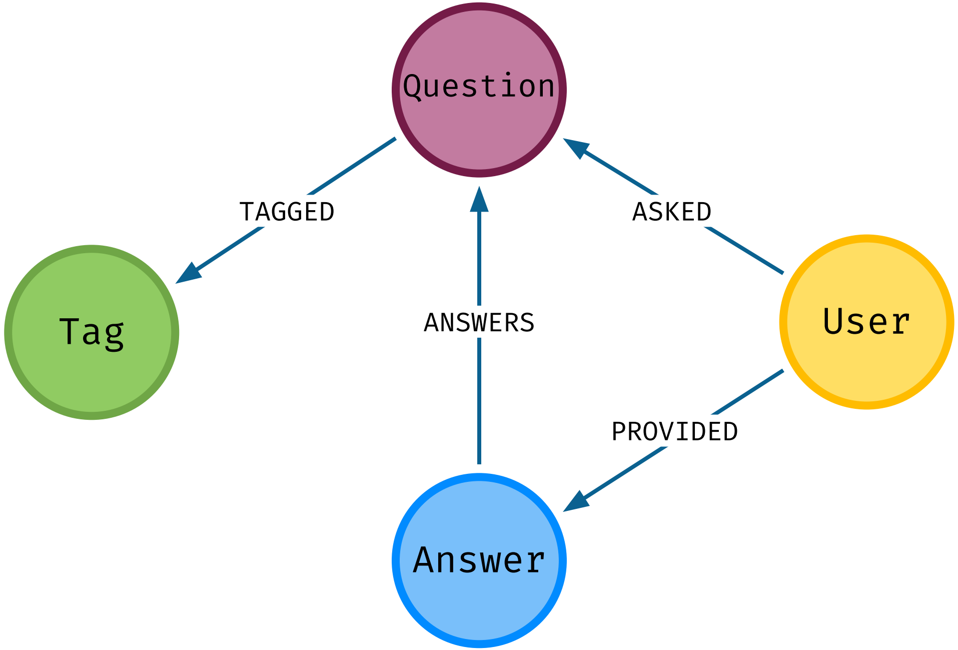
Data starts with things. In StackOverflow those are...



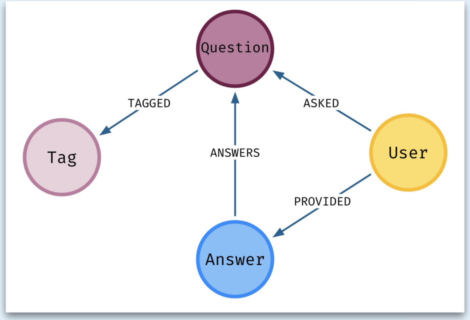
Data records information about things

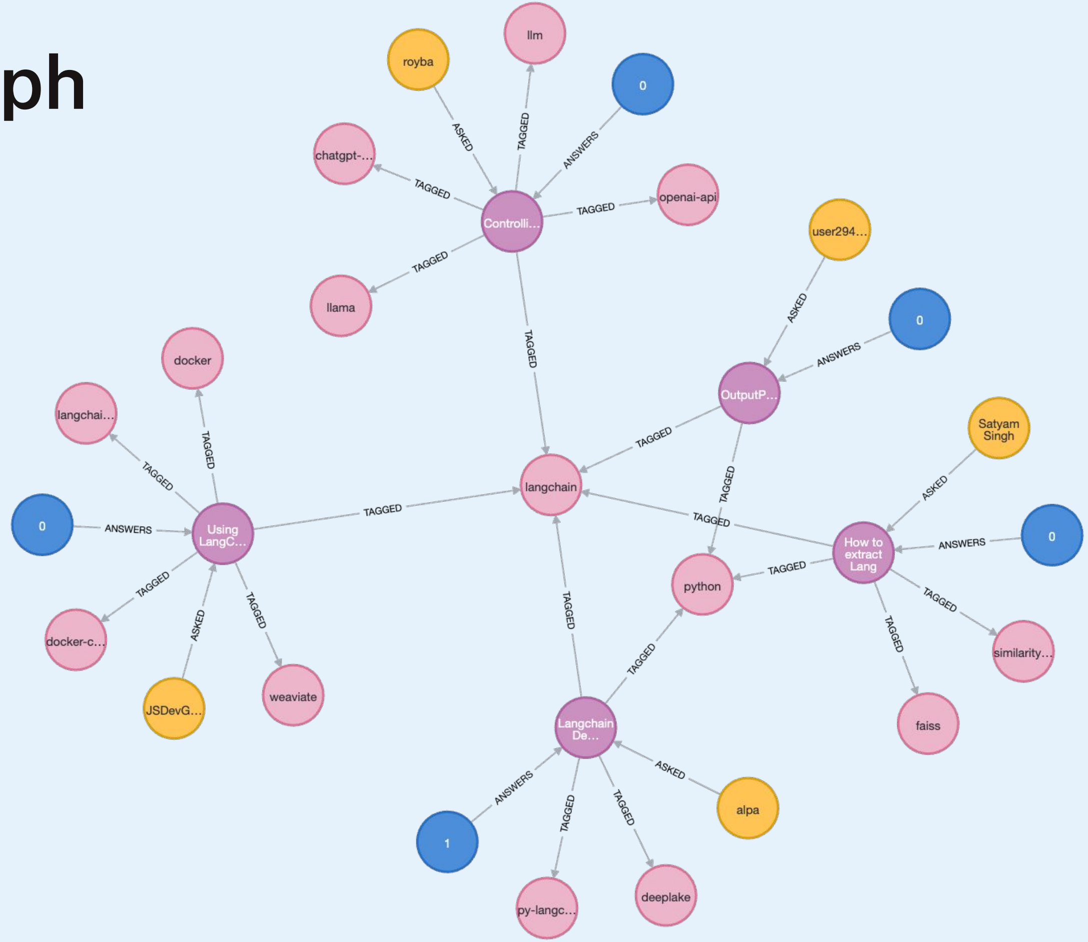


Data records can be related

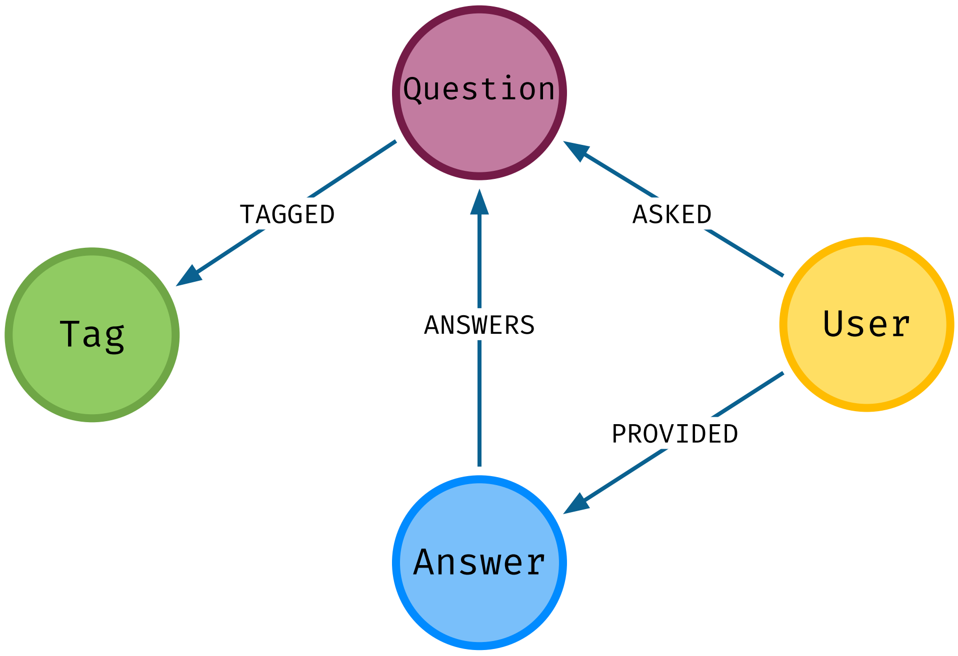


StackOverflow Knowledge Graph

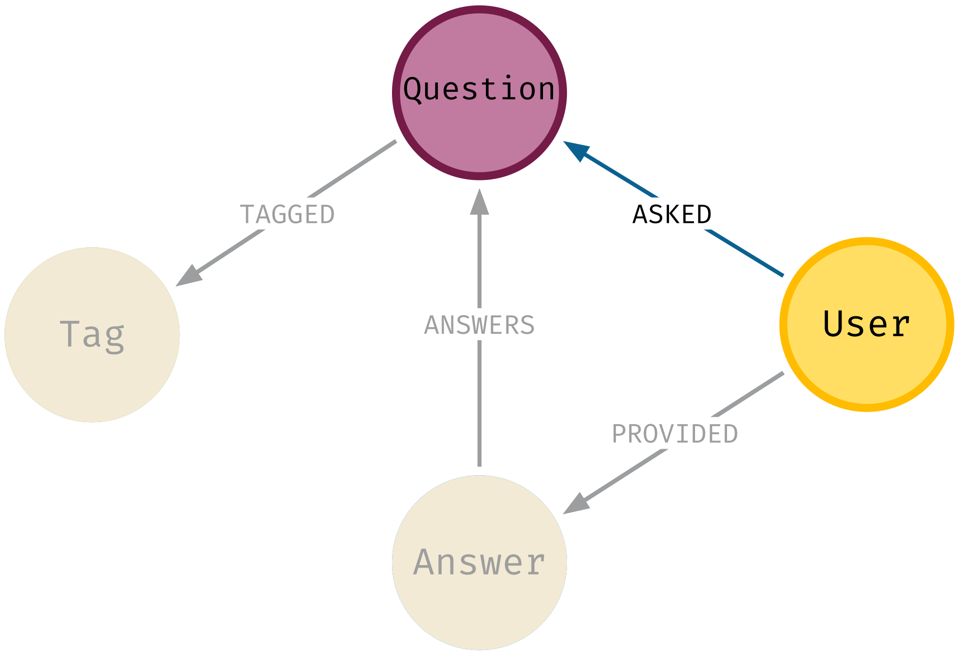




Data relationships create patterns

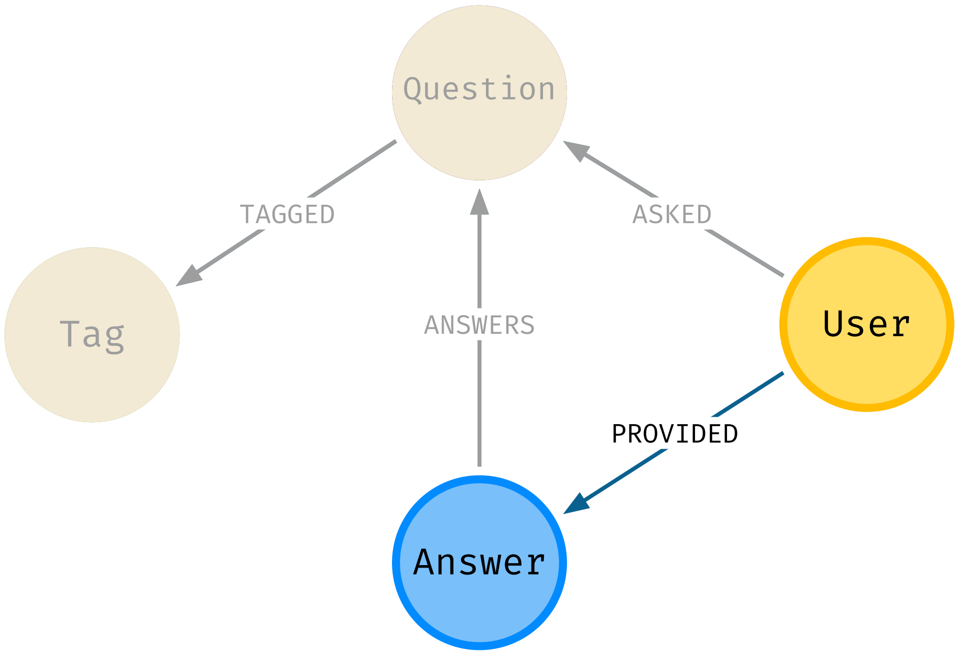


Data pattern: from users to questions



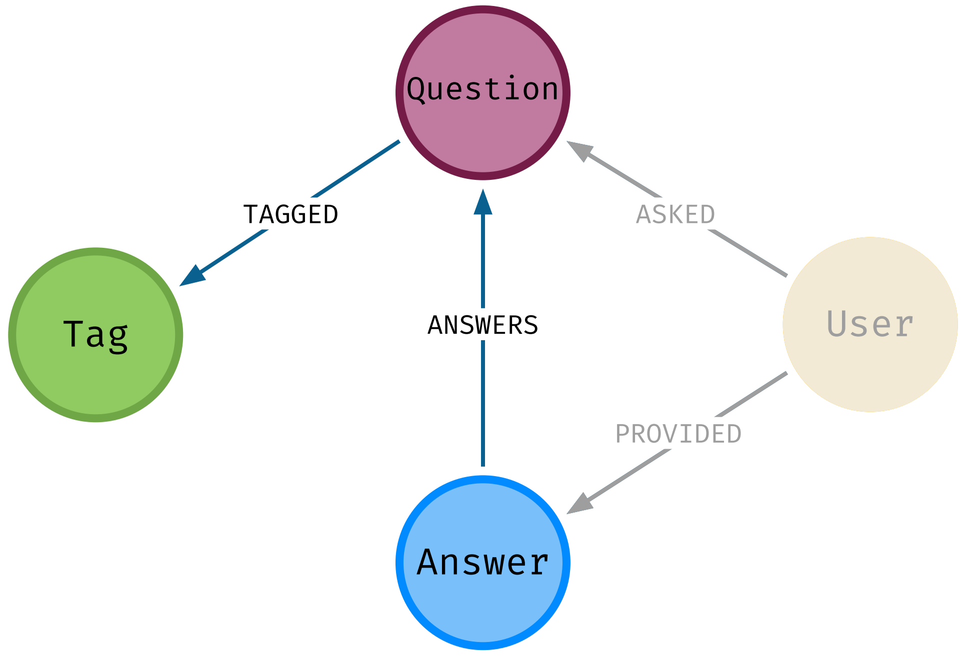
`(User)-[ASKED]→(Question)`

Data pattern: from users to answers



(User)-[PROVIDED]→(Answer)

Data pattern: from answers to tagged questions



(Answer)-[ANSWERS]→(Question)-[TAGGED]→(Tag)

Knowledge Graph

Facts about people, places, or things **interlinked** by their relationships

Human and LLM-friendly readable format

Organizing principle provides context for reasoning about the data

Making it Real

GenAI Apps in the Enterprise

understand customer behavior and preferences better, to provide personalized services.

Recommendations

Customer Service

process documents to accelerate and reduce the effort to codify rules

quickly answer customer questions from thousands of pages of policy documentation

Supply Chain

identify bottlenecks in bill of materials to support demand for customers

product recall and associated quality control checking

Knowledge Base

patient portal transformation initiative

internal documentation search

answer prospect questions on the fly

chatbot which queries publications, news, etc. by predefined prompts.

Policy & Pricing

simplify and summarize resources to help technicians resolve errors on the factory floor.

identify the right coverage, with the least effort, at the best possible price.

Fraud Detection

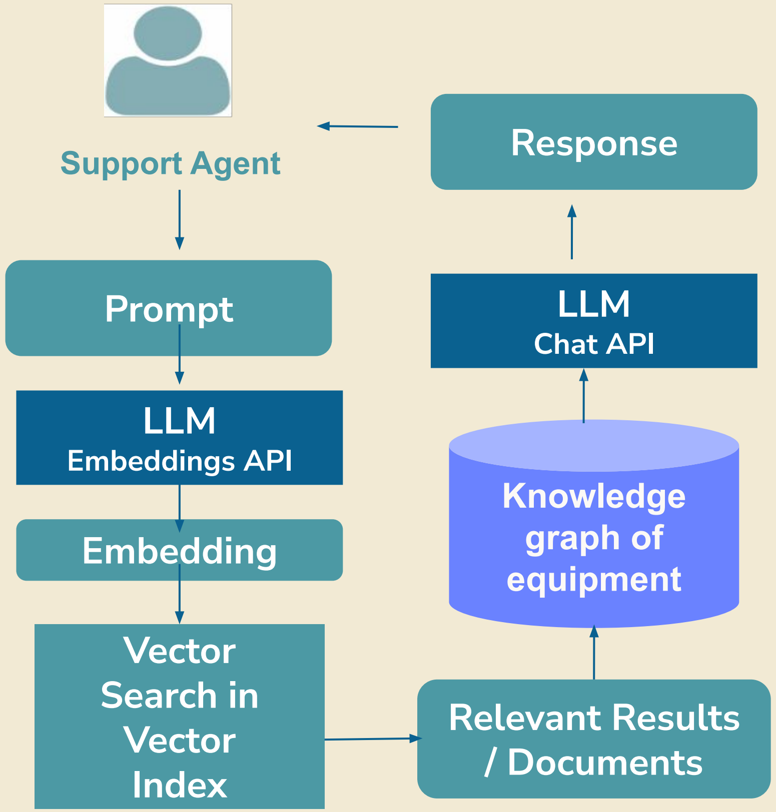
Improve Answers: Pure text w/ Knowledge Graph

Global Equipment Manufacturer:

Challenge: Support agents want to lower their mean time to repair (MTTR) by quickly providing a resolution, but the best answer is hidden among thousands of field docs, engineering summaries, documentation, and case histories.

Solution: Use vector embeddings to look for the most relevant information, while the Neo4j knowledge graph adds the exact match for critical components.

Desired Outcome: Reduce MTTR, and provide the best option for the customer quickly and with confidence.



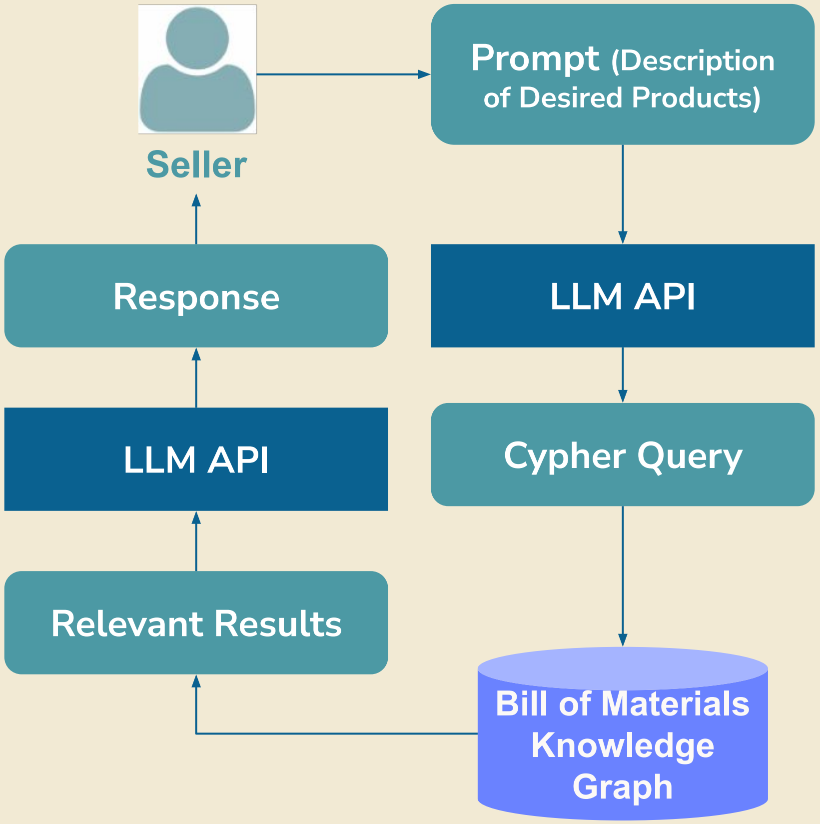
Knowledge Access: Pure data w/ Text to Query

Global Electronics Manufacturer:

Challenge: Sellers want to quote the right product combination and price for every opportunity across five unique types of external stakeholders. Information is stored across millions of bills of materials and product combinations.

Solution: Bring bills of materials into a knowledge graph that can be queried by an LLM interface by sellers.

Desired Outcome: Quote the best option for each customer quickly and with confidence.



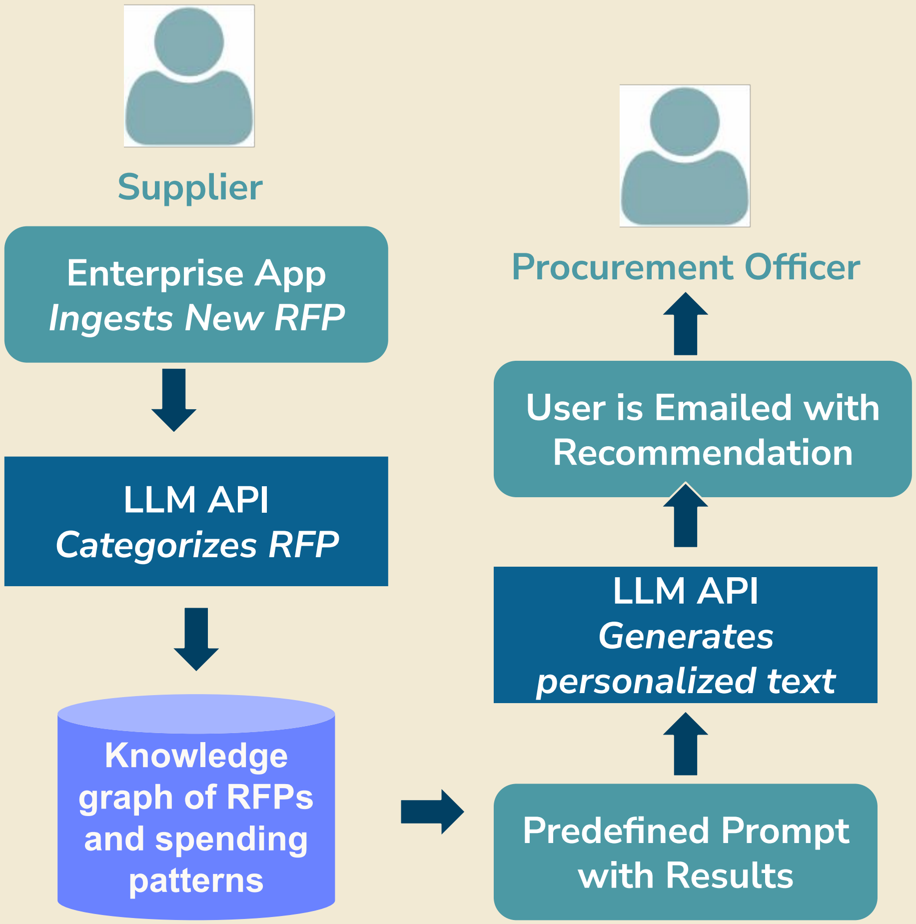
Complex Processes: Mixed Text + Data

Government Procurement Entity:

Challenge: Government entities can overspend or acquire goods and services redundantly because the volume of RFPs makes reviewing each of them resource prohibitive.

Solution: Use LLM to read the nature of RFP and classify it accordingly. Compare the new RFP against the knowledge graph of active and historic RFPs and associated spend. Recommend opportunities for consolidation or terms negotiation with suppliers.

Desired Outcome: Save tax dollars for other important projects



RAG is the way to bridge GenAI with Business Data

Lorem ipsum

Thanks!

@akollegger most places
andreas@neo4j.com