Who Am I?

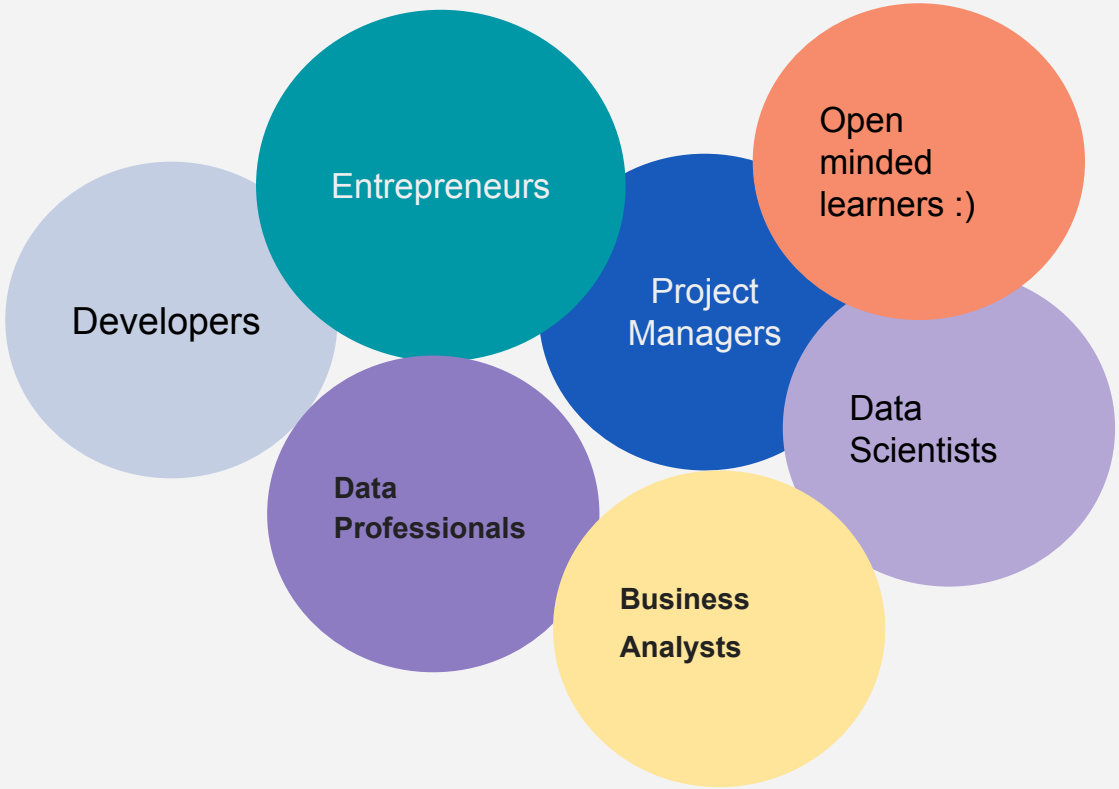
Paulo Dichone

Software, Cloud, AI Engineer
and Instructor

What Is This Course About?

- Knowledge Graphs & RAG -
  - Fundamentals of Knowledge Graphs
  - Implement Knowledge Graphs to Improve RAG Systems
  - Optimization and Best Practices

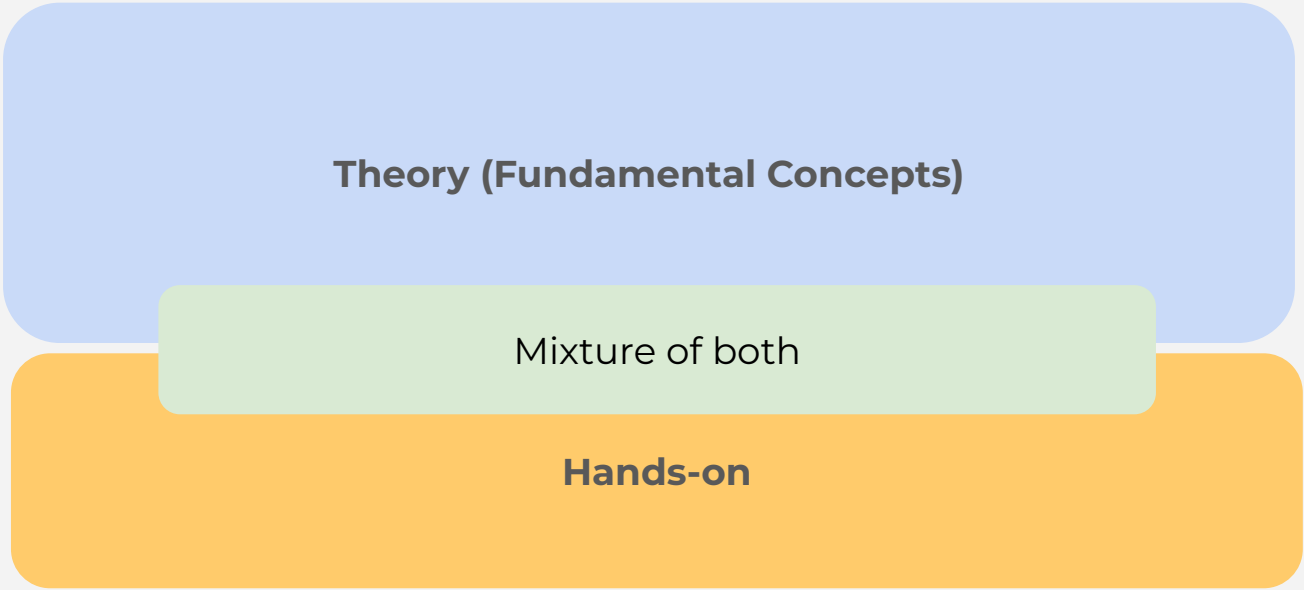
Who Is This Course For



Course Prerequisites

1. Know Programming (highly *preferred... at least the basics)*
    a. *There will be Python code*
    b. *Basics of LangChain, LLM, AI*
2. *This is not a programming course*
3. Willingness to learn :)

Course Structure



Development Environment setup

- Python
- VS Code (or any other code editor)
- OpenAI API Account and API Key

Set up OpenAI API Account

*** Please note** that you will need an API key to use OpenAI services, and there may be some costs associated with using the API. However, these costs should be minimal.

OpenAI API - Dev Environment Setup

Python (Win, Mac, Linux)

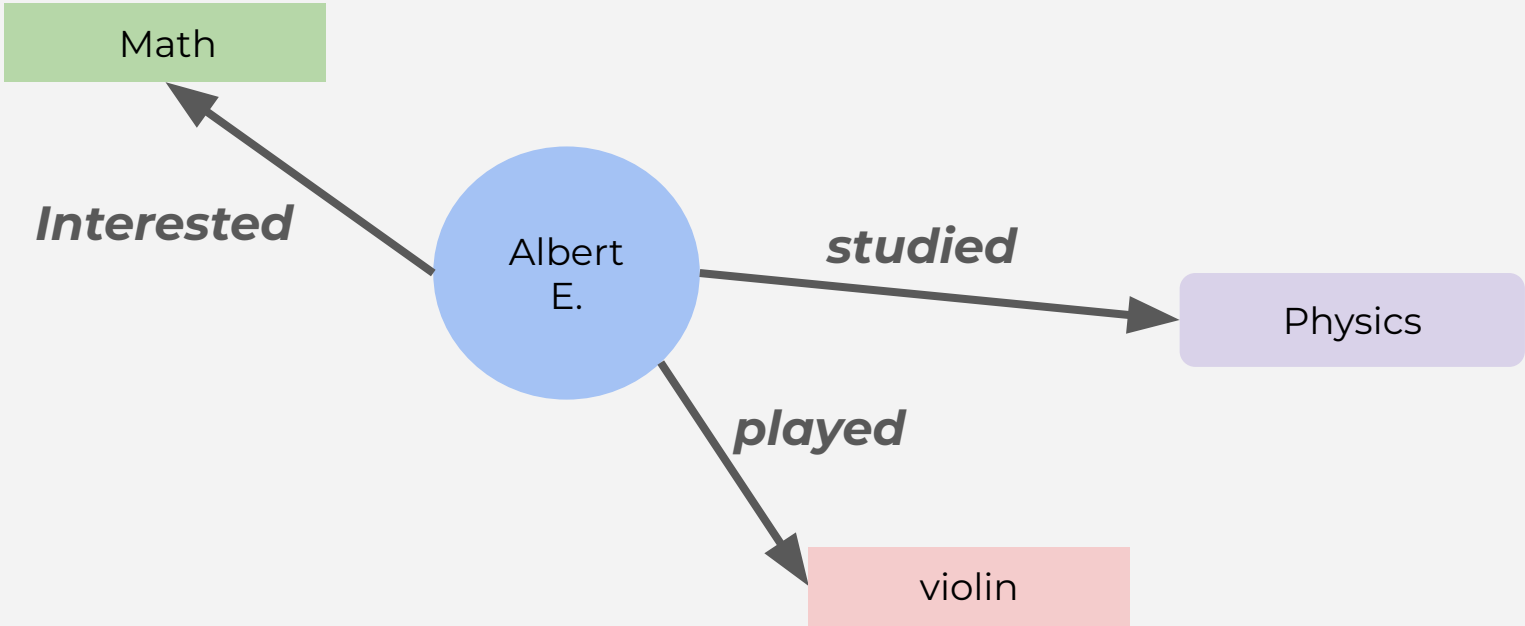
https://kinsta.com/knowledgebase/install-python/

Knowledge Graphs Deep Dive

- What are they?
- Why (motivation)?
- Advantages
- Key concepts
- Structure
- Components
- Use cases
- Advantages & Challenges

Knowledge Graphs

A KG is a **structured** representation of **facts** and **relationships** within a specific **domain...**



Key concepts

Key concepts

Attributes:

- Birth data
- Nationality
- profession

Entity

Edge/Relationship

Entity

Key concepts

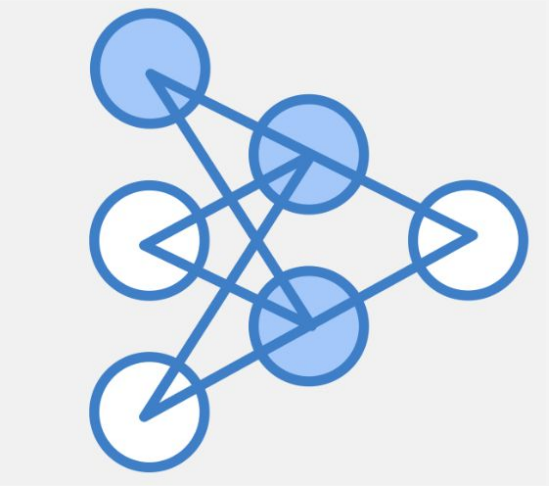
Attributes:

- Birth data
- Nationality
- profession

Triples

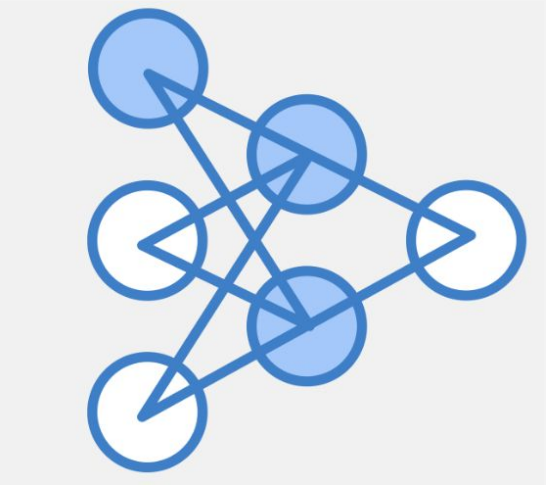
- Subject (Albert)
- Predicate (studied)
- Object (Physics)

Structure



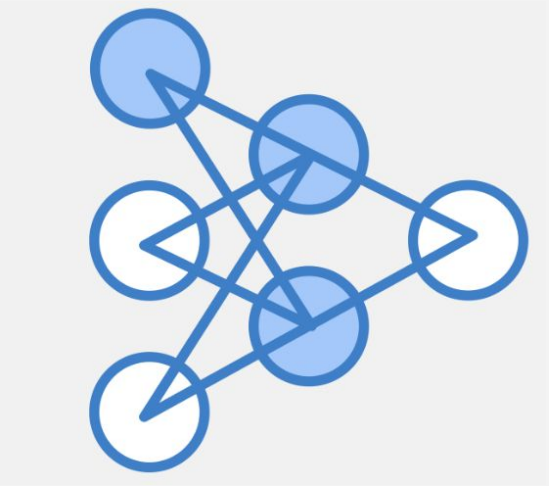
- **Graph-based Structure**: nodes, edges, to model real-world knowledge
- **Ontology**: schema that defines the types of entities, relationships and attributes
- **Taxonomy**: Hierarchical classification of entities in the graph

Structure



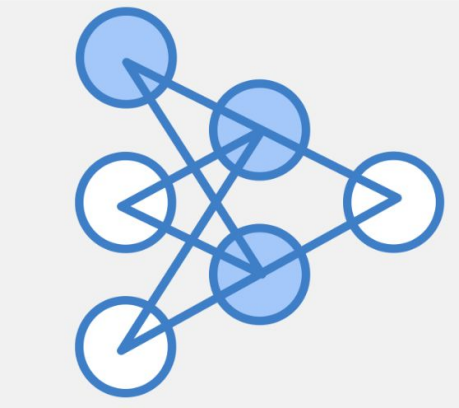
- **Nodes:** represent entities or instances
- **Edges:** relationships between entities
- **Labels:** tags assigned to nodes and edges to define their types
- **Properties:** additional information or attribute about nodes and edges

KG construction



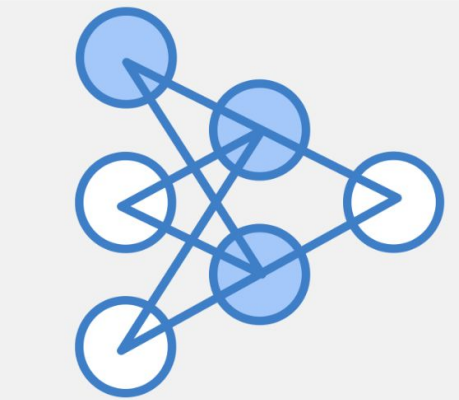
- **Manual curation:** manual input of data into graph
- **Automated Extraction:** using algos and NLP to extract information from unstructured data sources (txt, docs, DBs...)
- **Crowdsourcing:** input from a large number of sources to populate and refine the graph.

KG applications



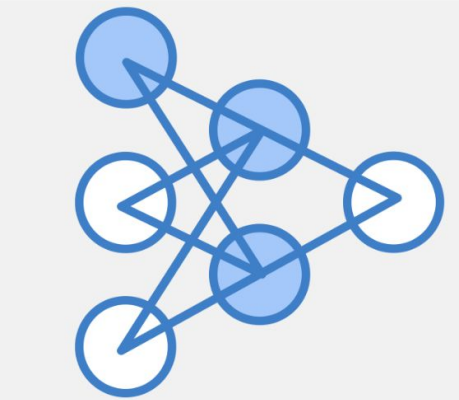
- **Search engines:** enhance search capabilities
- **Recommendation systems:** provide personalized recommendations
- **Data Integration:** combine data from various sources into one framework
- **AI and Machine Learning:** enhance machine understanding

KG Advantages



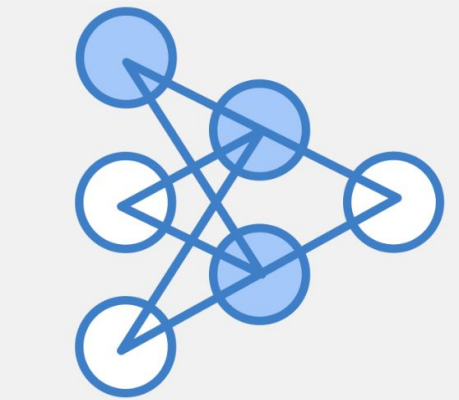
- **Enhanced data interoperability:** good for integrating heterogeneous data sources
- **Improved data quality and Consistency:** ensures accuracy (well-defined ontologies and taxonomies)
- **Facilitate Advanced Querying:** allows complex queries
- **Supports Inference and Reasoning:** machines can infer new knowledge from existing data easily

KG challenges



- **Data quality:** accuracy and reliability of the data is challenging
- **Scalability:** managing and processing large-scale KGs efficiently
- **Complexity:** developing and maintaining complex ontologies and taxonomies
- **Data Privacy:** securing sensitive data while utilizing KGs

Conclusion

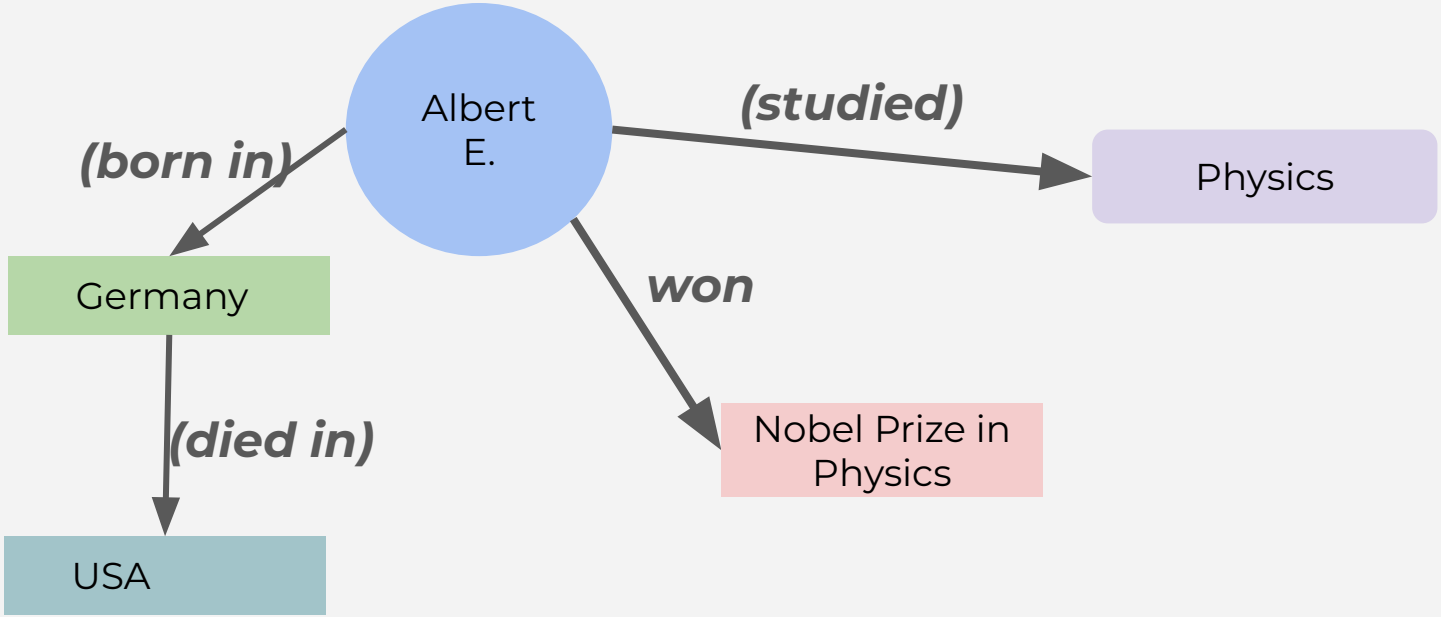


- **KGs:** a powerful tool for structuring and reasoning about complex data
- **Scalability:** enhance capabilities of AI systems (improve search and recommendation engines...)
- **Data integration:** facilitate data integration and analysis across various domains

Knowledge Graphs Deep Dive neo4j

- Building a Knowledge Graph
  - Introduction to neo4j
    - Relationships
    - Entities
- hands -on

Building a Knowledge Graph



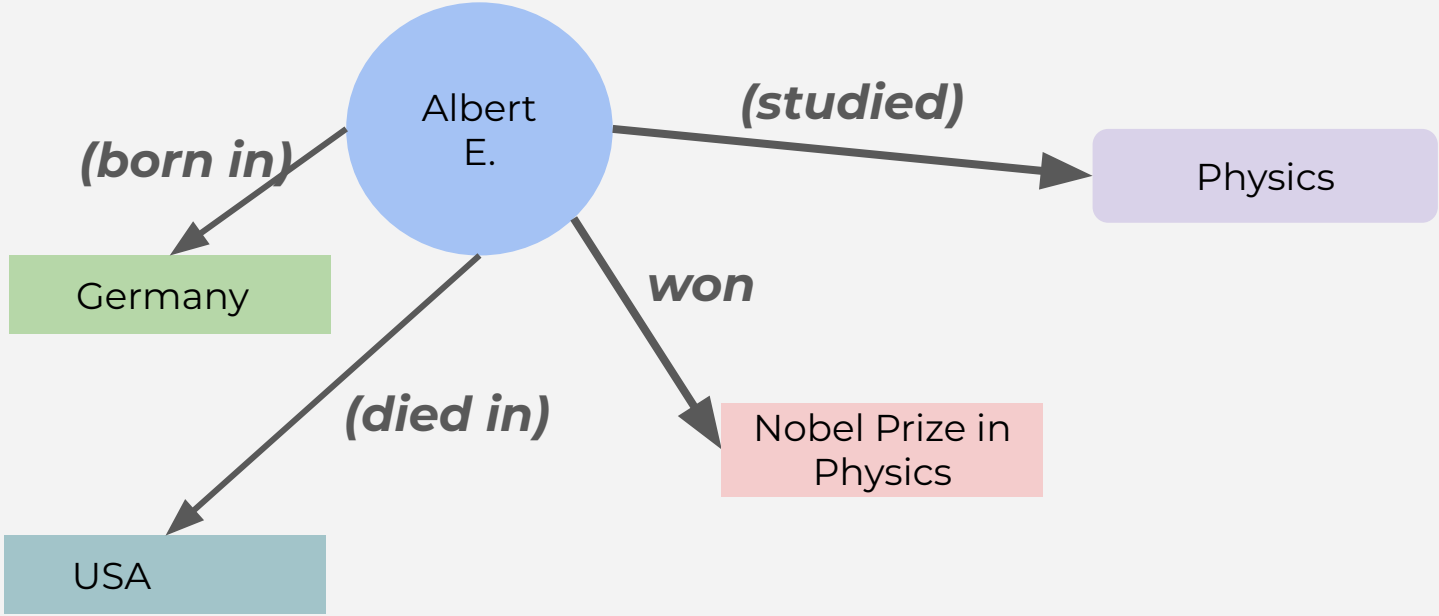
Technologies and Languages

RDF (Resource Description Framework)

SPARQL (SPARQL Protocol and RDF Query Language)

OWL (Web Ontology Language)

Neo4j



(a:Person)-[:STUDIED]->(p:Physics)

(a:Person)-[:WON]->(n:NobelPrize)

(a:Person)-[:DIED_IN]->(u:USA)

Neo4j graph database

Neo4j

A leading graph database management system that provides an ACID-compliant transactional backend for applications

Key concepts:

- Nodes
- Relationships
- Properties
- Labels
- **Cypher -** the query language for interacting with the graph data

Summary

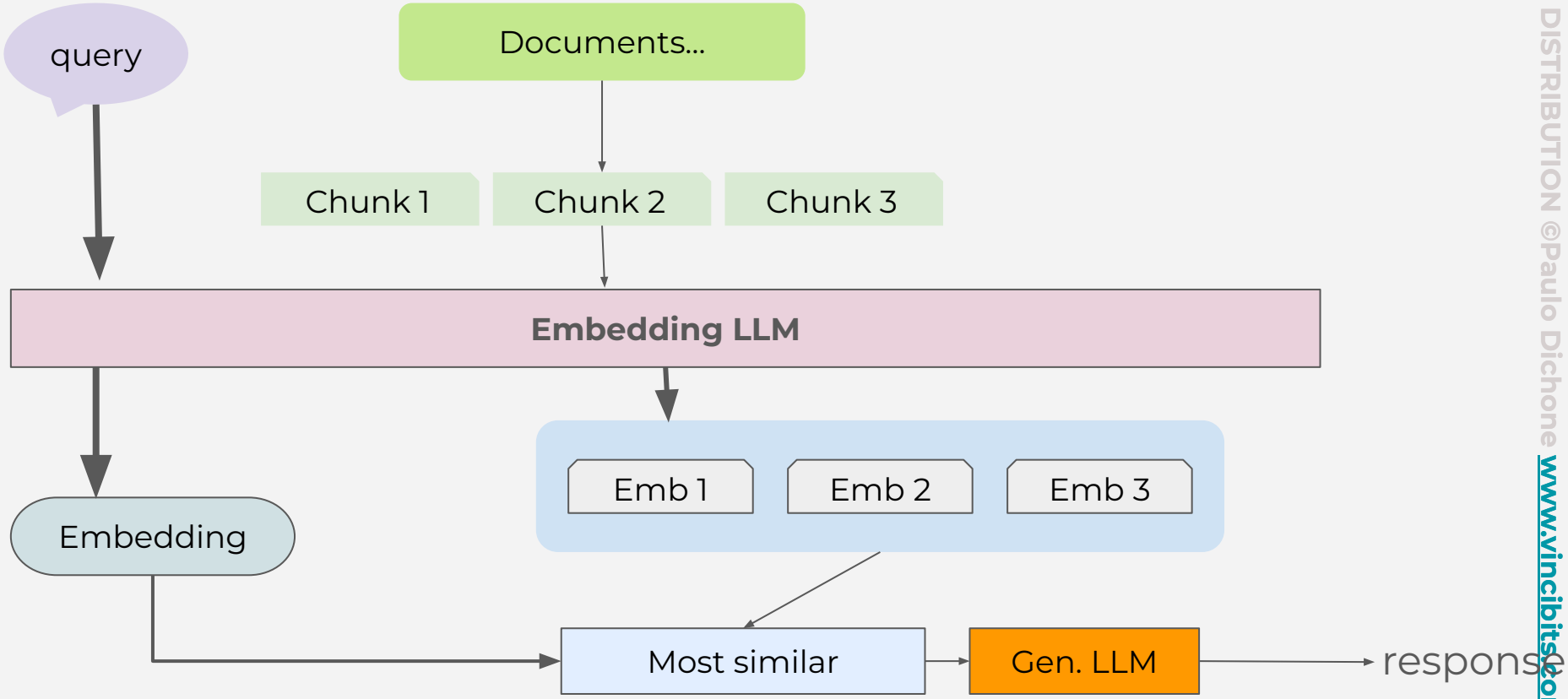
- Neo4j set up
  - Account
  - Create a graph database
  - Neo4j browser
  - Visual KGs
- Create a KG programmatically
- Querying the KG

Knowledge Graphs & RAG

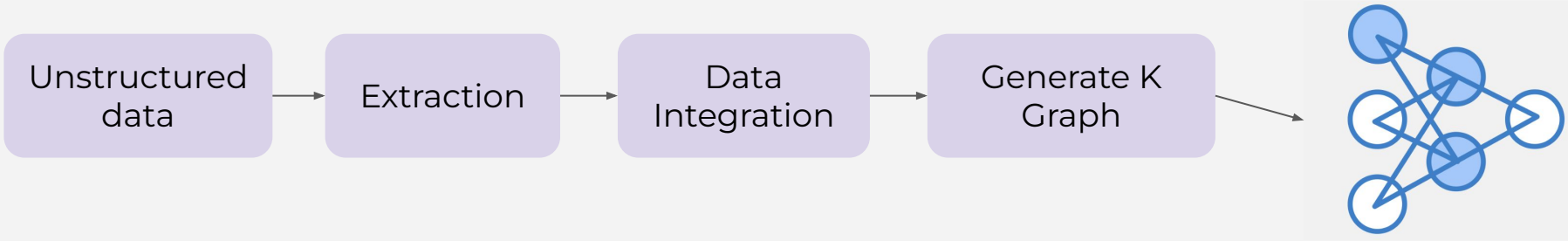
Graph retrieval-augmented generation (GraphRAG)

A powerful addition to traditional vector search retrieval methods

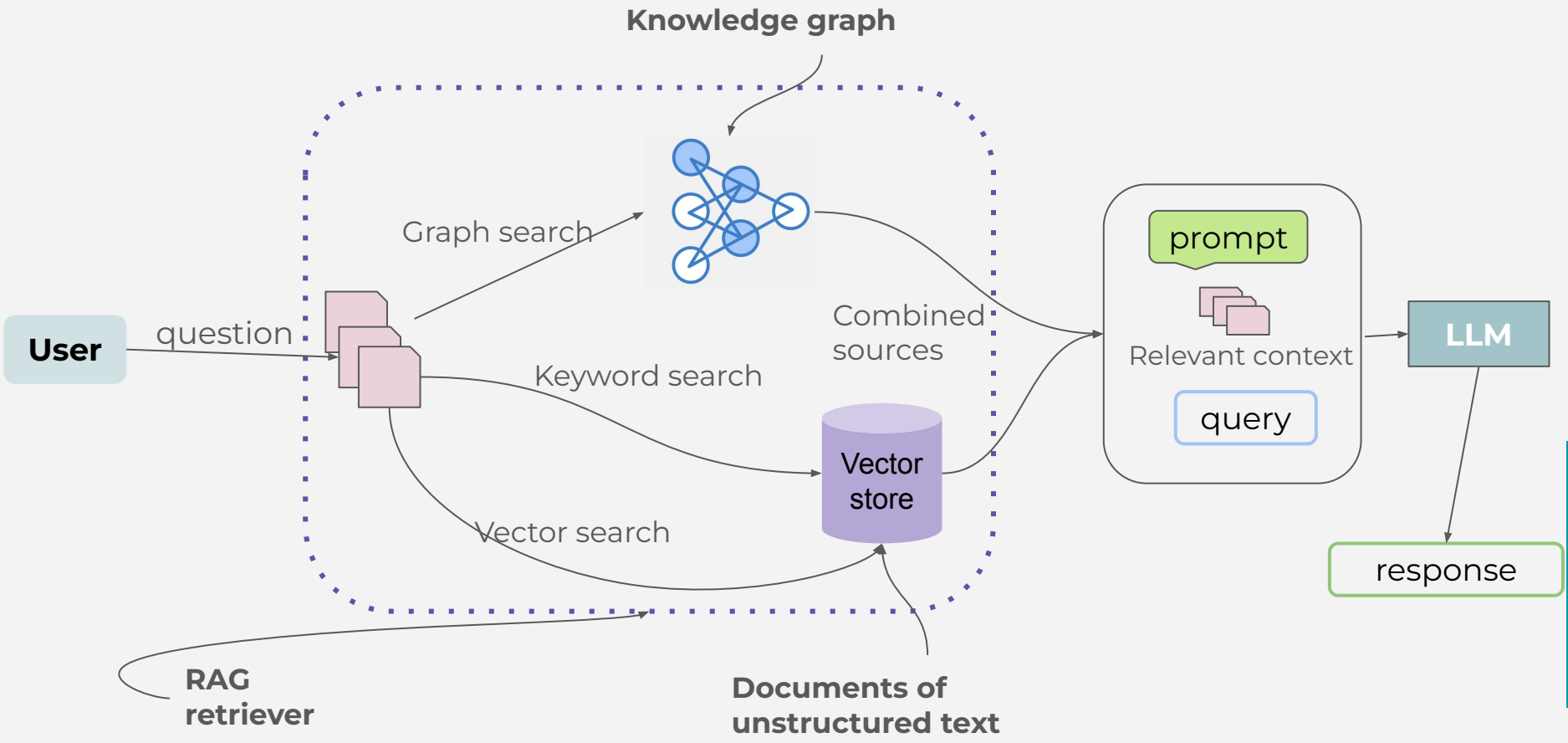
RAG Overview



Knowledge Graph & RAG - Steps



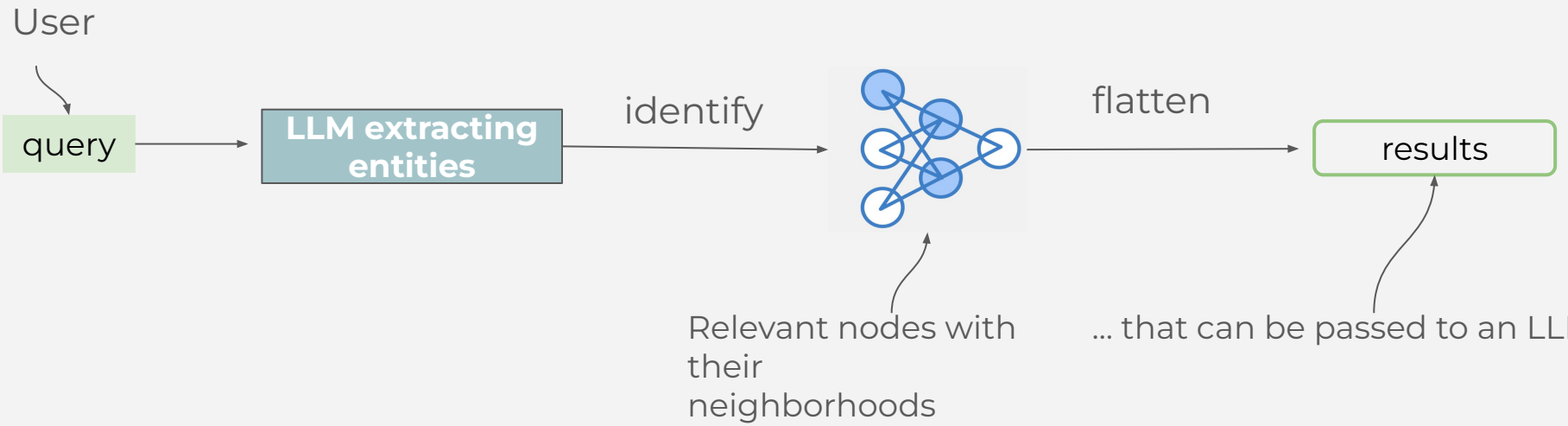
Knowledge Graph & RAG - Overview



GraphRAG
Hand-on

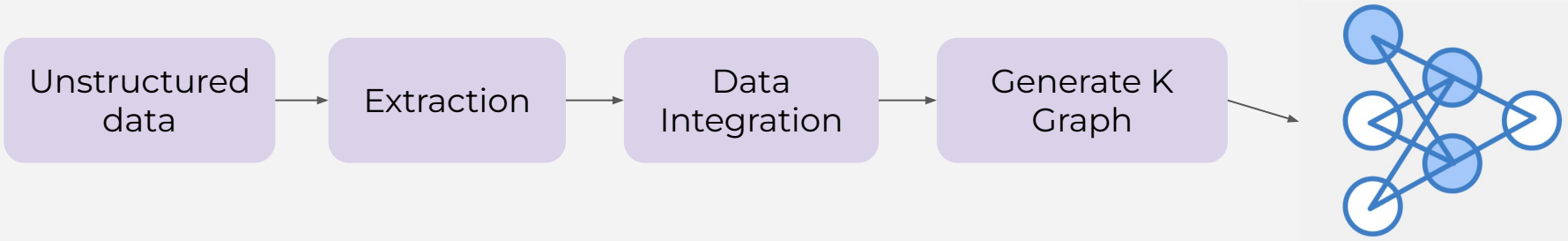
Build a Knowledge Graph and a Hybrid Retrieval for RAG

Graph Retriever - Flow



Hands-on - The Roman Empire RAG System

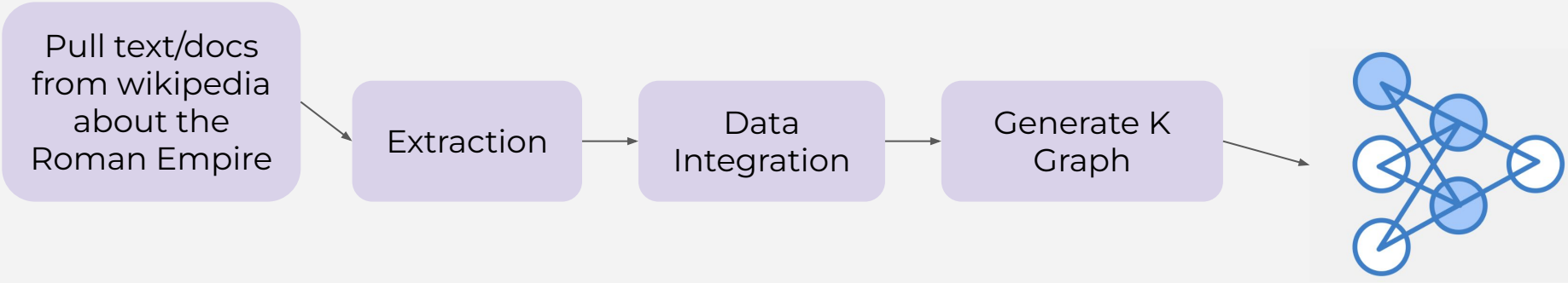
Enhance the accuracy of RAG application with Knowledge Graphs



- Unstructured data retrieval (vector store, keyword search, vector search)
- Graph Retrieval (more involved, but offers more freedom)

Hands-on - The Roman Empire RAG System

Enhance the accuracy of RAG application with Knowledge Graphs



Congratulations!

You made it to the end!

- Next steps...

Course Summary

- Knowledge Graphs for RAG (GraphRAG)
  - Fundamentals of Knowledge Graphs
  - Key characteristics
  - How they work
  - Deep dive into Building Knowledge Graphs
    - Built first simple KG
    - Basics Cypher Query Language
    - The Neo4J Graph database
    - GraphRAG Hands on
      - Using framework to extract, embed unstructured text
      - Full on real-world use case scenarios

Wrap up - Where to Go From Here?

- Keep learning
  - Extend the projects we worked on in this course
  - Implement your own knowledge Graphs and RAG
- Challenge yourself to keep learning new skills!

Thank you!