ORACLE

Graph RAG in DB

ML Summit 2024

Eduard Cuba (Oracle Labs)

Melliyal Annamalai (Spatial and Graph team, Oracle Database)

In this session

1. Lightning intro to Graph
2. Lightning intro to RAG
3. **Why Graph RAG**
4. **What is Graph RAG**
5. **How to implement Graph RAG** (Langchain, Oracle DB)
6. What's next for Graph RAG

Who are we

Melli Annamalai
Distinguished Product Manager

Nashua, NH

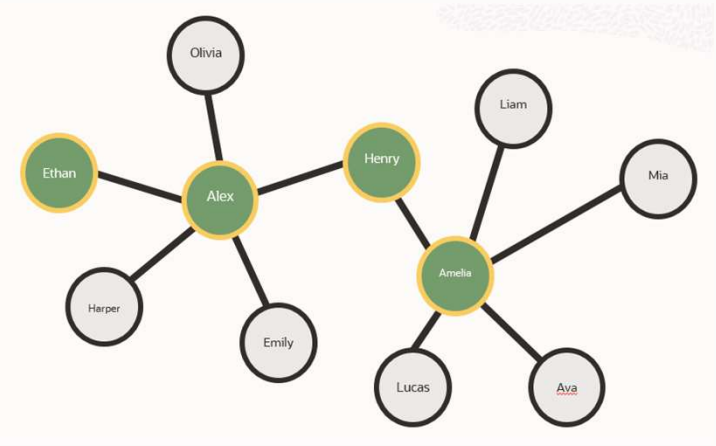
Eduard Cuba
Senior Member of Technical Staff

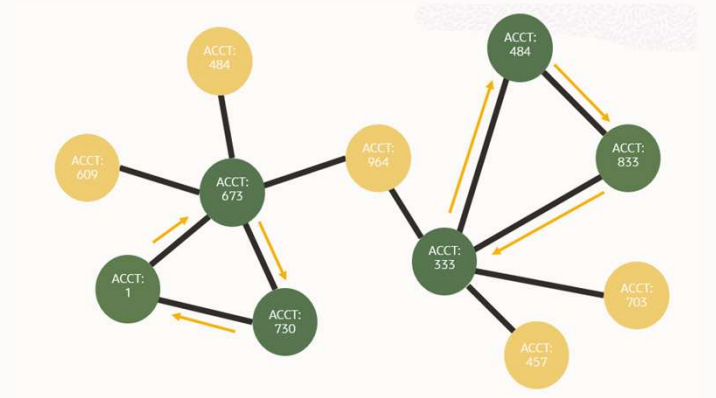
Zürich, Switzerland

Safe harbor statement

The following is intended to outline our general product direction. It is intended for information purposes only, and may not be incorporated into any contract. It is not a commitment to deliver any material, code, or functionality, and should not be relied upon in making purchasing decisions. The development, release, and timing of any features or functionality described for Oracle's products may change and remains at the sole discretion of Oracle Corporation.

What is a Graph?





Entities represented as **nodes/vertices**

Relationships between them represented as **edges**

Examples

- Social media: How are people connected?
- Bank transactions: What are anomalous patterns?
- Manufacturing: What are dependent components?

Graph use cases are part of many industries

Financial Services

- Find anomalous patterns
  - Detect money laundering
  - Uncover Fraud

Manufacturing

- Components and sub-components dependency analysis
- Supply chain analysis
- Digital thread connecting related components

Find Communities

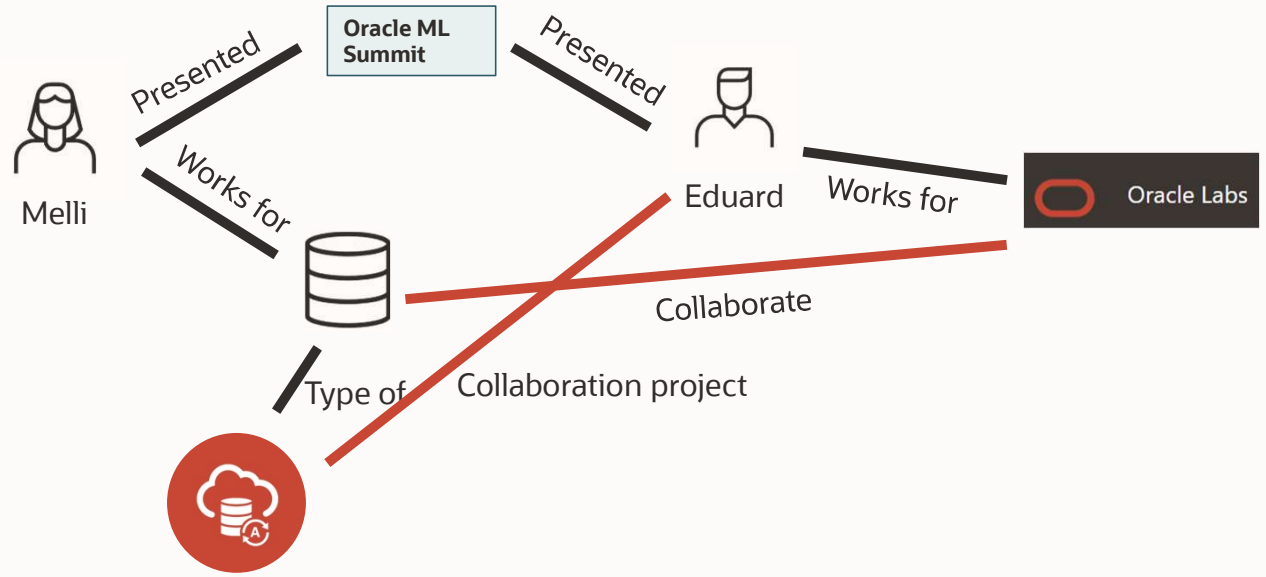
- Healthcare
  - Find patient communities
- Retail
  - Cluster customers based on connections to other customers and products
  - For cross-sell and upsell

Analyze Networks

- Telecommunications
- Transportation
  - What-if analysis
  - Root cause analysis of failures
  - Re-routing during outage

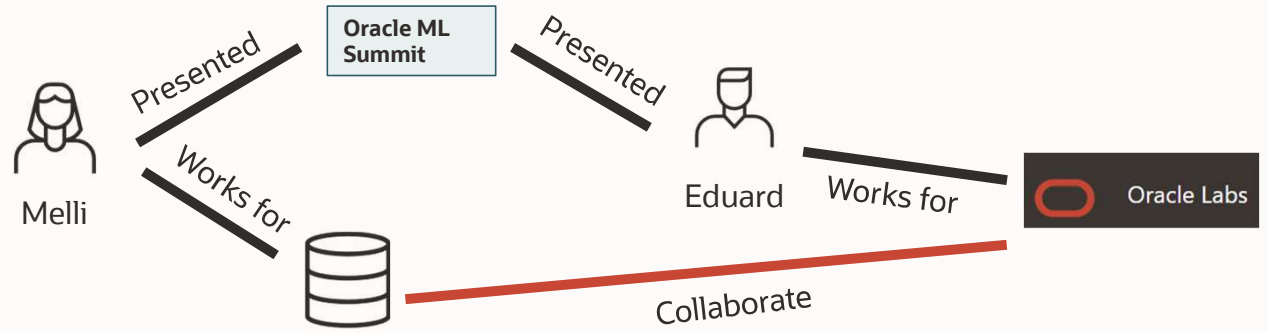
Graphs and Generative AI

Knowledge graphs can capture complex relationships in the real world



Graphs and Generative AI

Knowledge graphs can capture complex relationships in the real world



Graphs can help ground data provided to an LLM

- Provide *facts* to generative AI, with nuances and complexities captured in a knowledge graph

Developers are looking for ways to improve the quality of results from an LLM

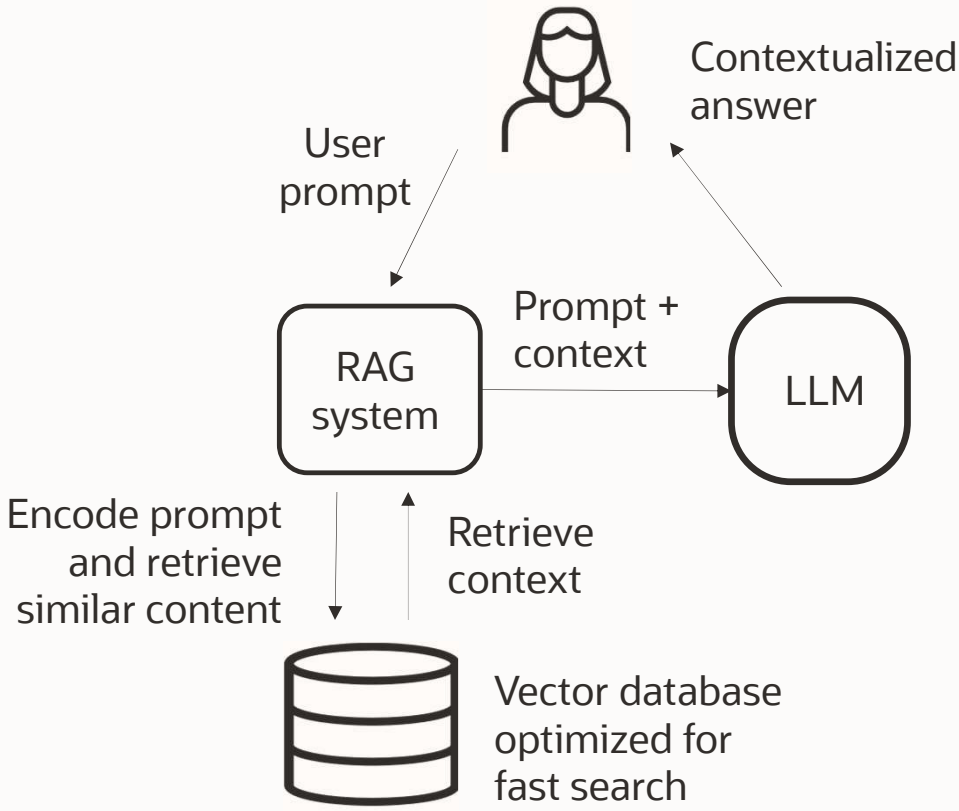
- A very active topic of research
- **Can graphs help?**

Lightning intro to RAG

What is a RAG?

Retrieval Augmented Generation in a nutshell

- **Goal:** Use your data as **context to the LLM**
- Split data into small pieces (chunks)
- Create numerical descriptions of the chunks and store them in a vector database
- Find best matching pieces of information related to the query
- Provide top matches as context to the LLM



RAG in Oracle DB

So … RAG is an LLM with a Vector Index?

Easy in Oracle DB!

```
DBMS_CLOUD_AI.CREATE_VECTOR_INDEX(
   index_name => 'VECTOR_RAG',
   attributes => '{
      "vector_db_provider": "oracle",
      "location": "https://oci.com/my_text_file",
      "object_storage_credential_name": "OCI_CRED",
      "profile_name": "GENAI",
      "vector_dimension": 384,
      "vector_distance_metric": "cosine"
   }'
);

SELECT AI narrate 'What's Oracle Labs up to these days?';
```

Or use OracleVS in Langchain

RAG in Oracle DB: Example

Let's bring in an example, and let's make it a bit confusing – with a **Sherlock Holmes story**!

The Adventure of the Blue Carbuncle story

TLDR: **James Ryder,** the waiting maid **steals a jewel** from a guest **and frames John Horner**, a plumber with a criminal record.

```
DBMS_CLOUD_AI.CREATE_VECTOR_INDEX(..., "location": "story.txt", ...);

SELECT AI narrate 'Who stole the jewel?';

> According to the context, John Horner, a 26-year-old plumber, was accused of having abstracted the gem known as the blue carbuncle from the jewel-case of the Countess of Morcar.
```

Correct, but doesn't answer the question!

RAG in Oracle DB: Example

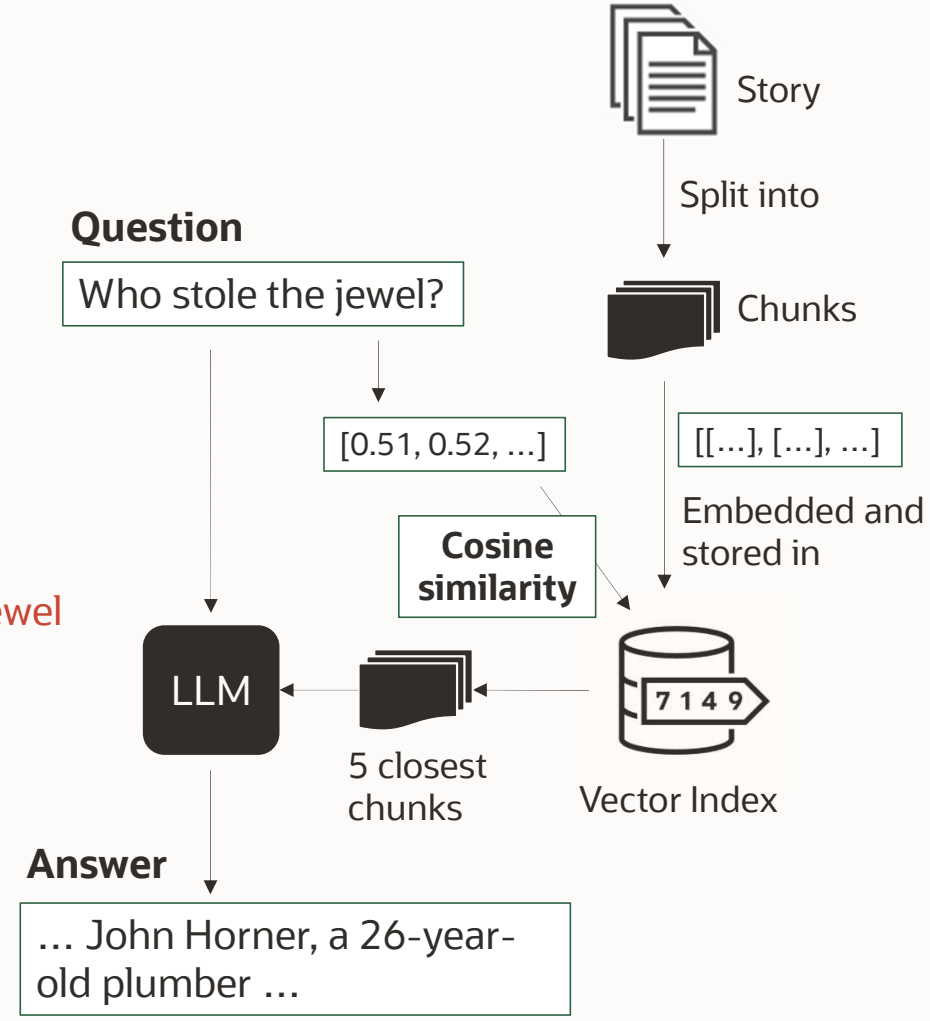
What went wrong?

Need to understand what how RAG works internally

- Story split into chunks
- Chunks embedded and stored in Vector Index
- Closest chunks retrieved as context

The problem

- The text **does not explicitly say**: **Ryder** stole the jewel
- One has to understand the story
- Vector Index **does not understand the data**

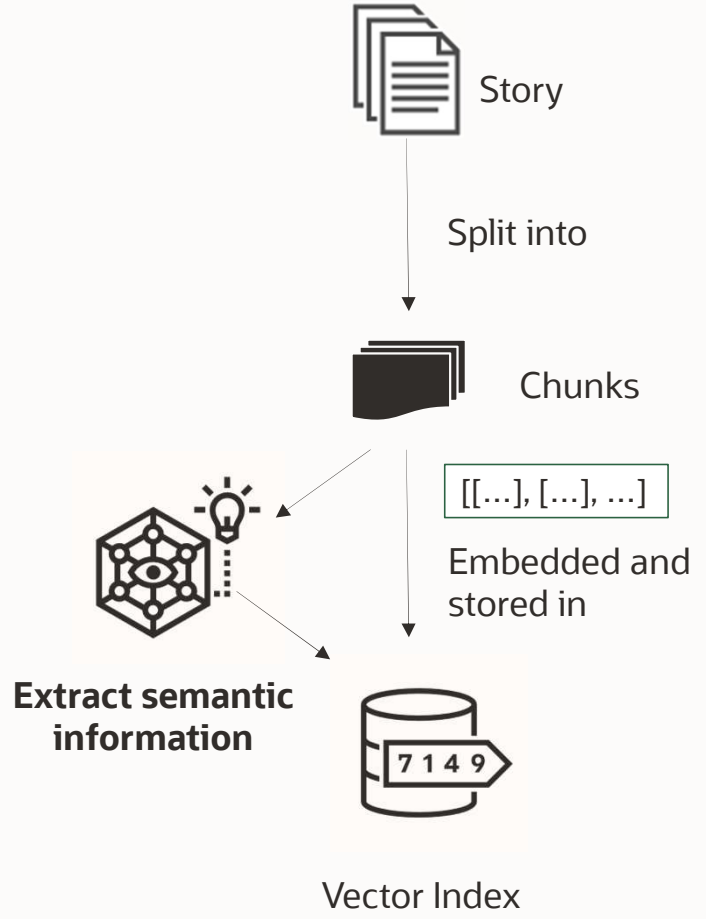


Idea: Extract Information from the data

What if ... Instead of "*just chunking the data*"

Extract semantic information from the chunks
- "*take notes while reading the story*"
- Try to understand the data

Different ideas how to approach this
- Classical NER based Information Extraction
- Contextual Retrieval (Anthropic blog post)
- Graph RAG (Microsoft paper, Neo4j blog)
  - Different flavors
  - Mostly using LLMs these days
    - Extract entities and relationship
    - Build a Knowledge Graph
    - Optional: Generate summaries hierarchically



Why Graphs?

How about Graphs?

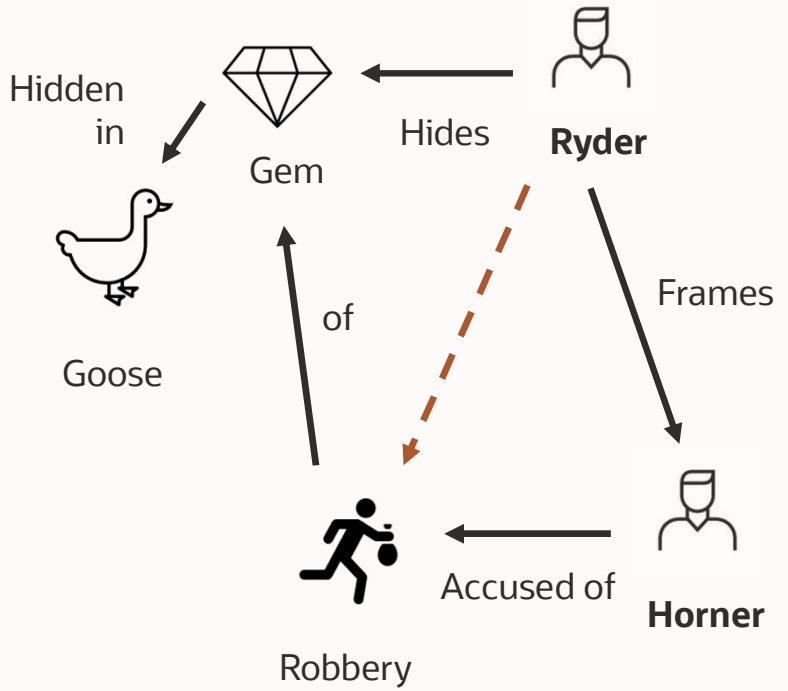
- **Great for capturing relationships** in the data
- **Represent data in a structured way**
  - Updatable, auditable, transparent

Use Graphs to improve the precision of the Retrieval

- **Connect information** spread over a large data-base
- **Bring transparency** to the retrieval process

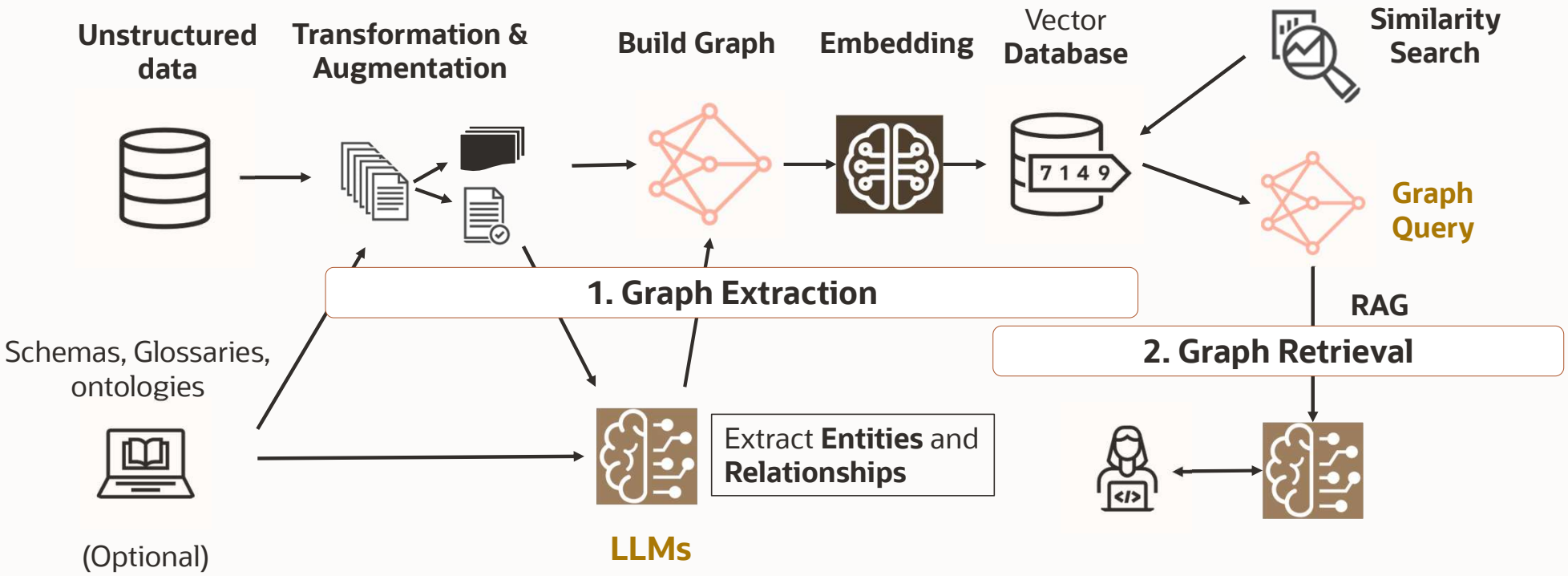
Already using Oracle Database?

- Your data is already in relational form!
  - But how do I bring the relationships into the RAG pipeline?



How do I get this though? … LLMs!

Using Graphs for RAG



1. Extracting a Graph

First step: Extracting a Graph from unstructured documents

What are the pre-requisites?

Extracting the Graph:
- Great out-of-the-box graph extraction support in 3rd party frameworks
  - Langchain LLMGraphTransformer
  - LlamaIndex PropertyGraphIndex

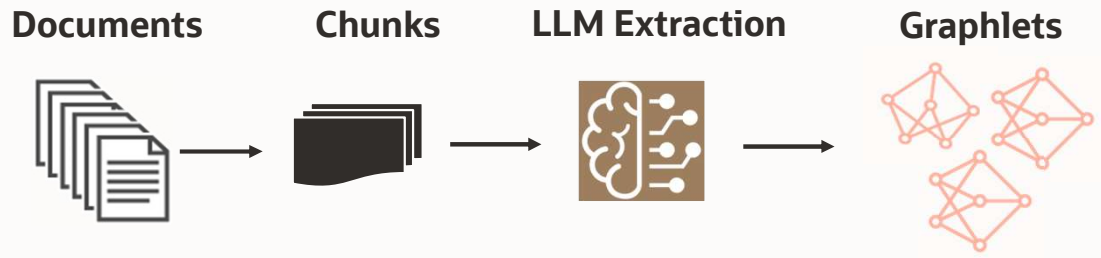
Having an access to an LLM
- Langchain natively supports OCI Gen AI

Storing the Graph:
- Langchain natively supports Oracle Vector Store
  - Including DBMS_VECTOR_CHAIN APIs

All we need! Let's put it together!

1. Extracting a Graph

Langchain example:
- Using Oracle in DB APIs



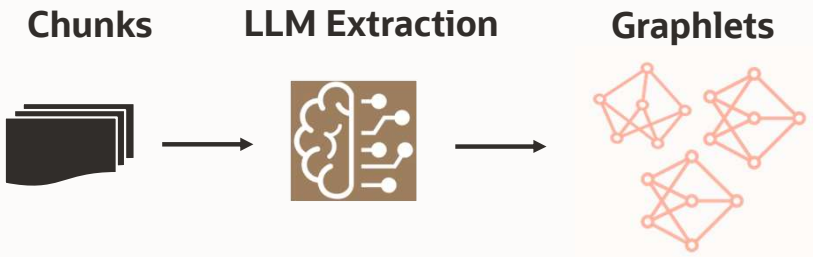

```
# chunk the documents
docs = OracleDocLoader(...).load()                          # dbms_vector_chain.utl_to_text
splitter = OracleTextSplitter(...)                  # dbms_vector_chain.utl_to_chunks
chunks = [Document(chunk) for chunk in splitter.split_text(docs)]

# configure the LLM and Graph Transformer
llm = ChatOCIGenAI(model_id="meta.llama-3.1-70b-instruct", ...)
llm_transformer = LLMGraphTransformer(llm=llm)

# extract a small graph from each chunk
graphlets = llm_transformer.convert_to_graph_documents(doc_chunks)
```

Why graphlets? – we get one small graph per chunks, they are not yet connected!

1. Extracting a Graph



How does the extraction work?

This is simply an LLM prompt applied to each chunk!

Prompt: You are a top-tier algorithm designed for extracting information in structured formats to build a knowledge graph.

Your task is to identify the entities and relations requested with the user prompt from a given text. You must **generate the output in a JSON format** containing a list with JSON objects.

Each object should have the keys: "**head**", "**head_type**", "**relation**", "**tail**", and "**tail_type**" …

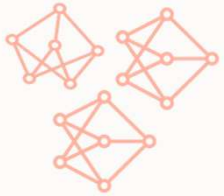
Default prompt from Langchain LLMGraphTransformer

1. Extracting a Graph

Exploring the graphlets

```
# extract a small graph from each chunk
graphlets = llm_transformer.convert_to_graph_documents(doc_chunks)
```

Graphlets



Chunk:

… It seems to me, **Ryder**, … knew that this man **Horner, the plumber**, had been concerned in some such matter before … Then, when he had left, **you rifled the jewel-case**, raised the alarm, **and had this unfortunate man arrested**. … you thought little enough of this poor Horner in the dock for a **crime of which he knew nothing**.

Output: Extracted Entities and Relationships

…
Node(id='**Ryder**', type='Person') -[**COMMITTED_CRIME**]-> Node(id='**theft**', type='Crime')
Node(id='**Ryder**', type='Person), -[**FRAMED**]-> Node(id='**Horner**', type='Person'),
…

1. Extracting a Graph

Merging the graphlets:

Problem: *Is **Sherlock Holmes** from Chunk 1 the same as **Sherlock Holmes** from Chunk 30? Is **Holmes** the same too?*

Simple solution: Assume all entities with the name and type are the same

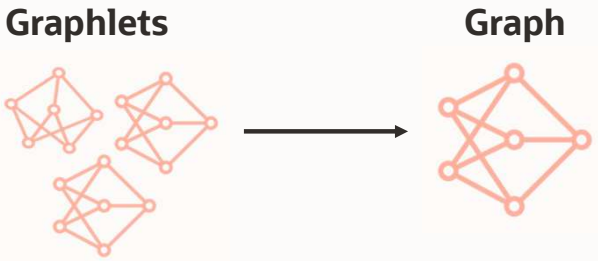
Caveats

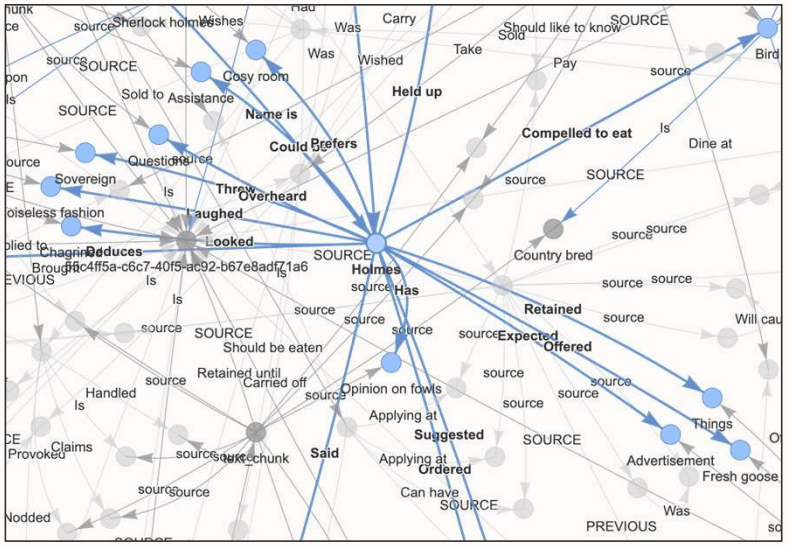
- Holmes != Sherlock Holmes **– no entity resolution**
- Joe from one customer file = Joe from another **– conflicts**

Functionally:

- **Store Relationships** in a Relationships table
- **Store unique Entities** in the Relationships in Entities table

For production consider Entity Resolution

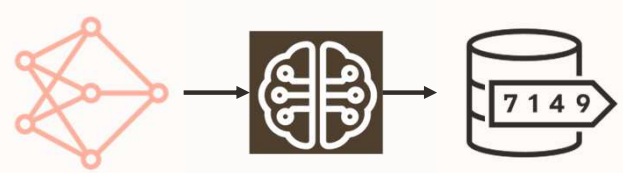




Holmes in the extracted graph

1. Storing the extracted Graph

Build Graph → Embedding → Vector Database



Create **embeddings** to enable semantic retrieval

OCIGenAIEmbeddings(…)

DBMS_VECTOR_CHAIN.UTL_TO_EMBEDDING(…)

Store Entities

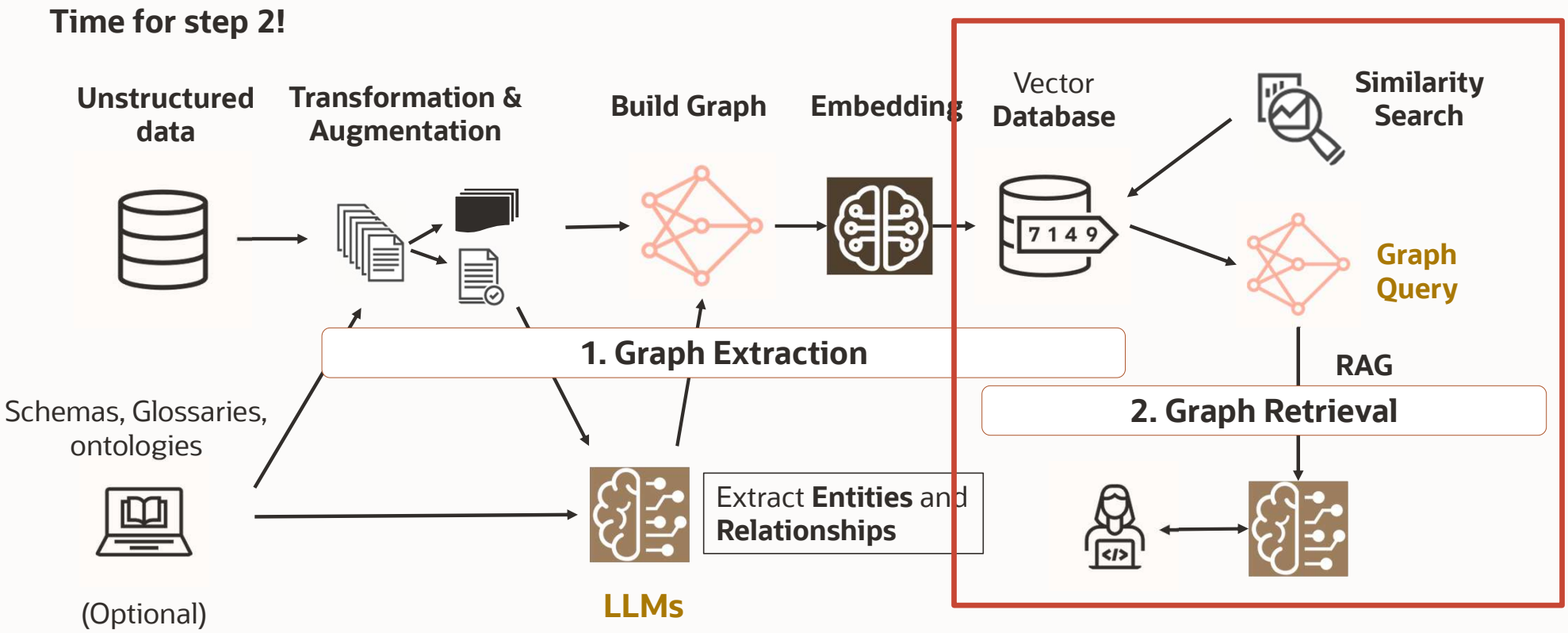
ENTITY_NAME	ENTITY_TYPE	EMBEDDING
Sherlock Holmes	Person	[-1.83563232E-002,2.8
purple dressing-gown	Object	[-6.48117065E-003,-1.
the shop window	Object	[2.17590332E-002,-2.5
hat-securer	Object	[-7.22045898E-002,9.2
man	Person	[-2.41661072E-003,6.3
hat	Object	[-3.51867676E-002,1.9
Peterson	Person	[-1.39923096E-002,8.6
Henry Bakers	Person	[-4.77294922E-002,8.1

Store Relationships

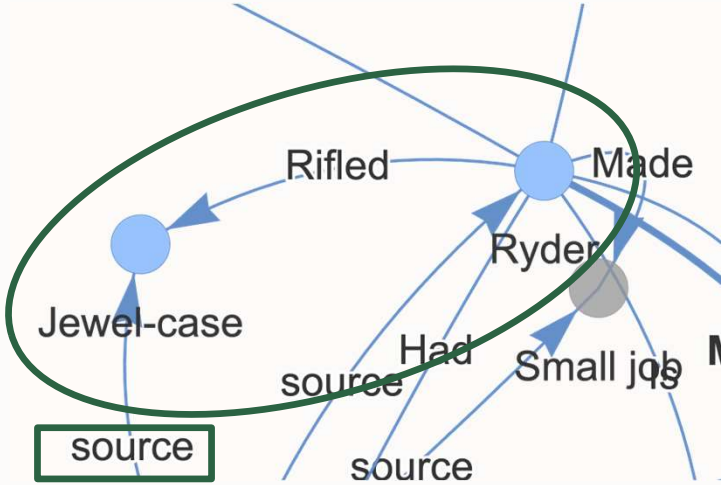
SOURCE	REL_LABEL	TARGET	DOC_ID	EMBEDDING
Person: Mr. Henry Baker	APPLY_AT	Location: 221B, Baker Street	D031A5D2B28D94F3	[4.33044434E-002,-2
Person: Holmes	EXAMINES	Object: stone	414C5ADD44378BB0	[2.73284912E-002,5.
Person: Peterson	WORKS_FOR	Person: Holmes	414C5ADD44378BB0	[1.04293823E-002,-4
Object: stone	FOUND_IN	Node: Amoy River	414C5ADD44378BB0	[1.40094757E-003,-7
Person: Holmes	SURMISED	Person: visitor		
Person: visitor	HAS_CHARACTERISTIC	Characteristic: man of learnin...	074B28DD59B0F150	[-8.36161641E-003,-
Person: visitor	HAS_CHARACTERISTIC	Node: ill-usage at the	074B28DD59B0F150	[-1.66168213E-002,-
Person: Horner	IS_INVESTIGATED_FOR	Crime: murders	A46CEB5A0D7BDFD4	[-1.37634277E-002,-

Embeddings (**source + label + target text**)

Using Graphs for RAG



2. Graph Retrieval strategies



Goal: provide context to the LLM using the extracted Graph

Many retrieval strategies exist

- MS Graph RAG
  - produce community summaries and use them as the context
- **Use the "right" chunk as the context** – how to find the **right** chunk?
  - Find related entities and take chunks of their relationships
  - Find text chunks and bring in related ones by traversing the relationships (+ PageRank)
  - **Find related relationships and take their chunks (via embedded triples)**
    - **We show this today**: e.g. get **source of (Ryder, Rifled, Jewel-case)**
- Generate a Graph query (not today, but maybe soon!)

2. Graph Retrieval strategies – find relevant relationships

What do I need to do?

- Find relationships similar to my query: VECTOR_DISTANCE - use Vector Index to make it fast!
- Retrieve the source text chunk - JOIN with text vector store

```
SELECT RELATION, TEXT FROM RELATIONS GR
        JOIN (SELECT ID, TEXT from TEXT_VS) VS ON GR.DOC_ID = VS.ID
ORDER BY VECTOR_DISTANCE(EMBEDDING,
        dbms_vector_chain.utl_to_embedding('Who stole the jewel?', ...)), COSINE)
FETCH APPROX FIRST 5 ROWS ONLY;
```

Correct answer in the first chunk!

RELATIONSHIP	TEXT
Person: Ryder -[COMMITTED_CRIME]-> Crime: theft	very scrupulous in the means you used. It s...
Person: Jem -[OBTAINED_INFO_FROM]-> Person: Maggie	is it you want, then?' "'That white one wit...
Person: John Horner -[ACCUSED_OF]-> Crime: Hotel...	market price." "A thousand pounds! Great Lo...
Person: Holmes -[FOLLOWED]-> Clue: clue	Holmes when he had closed the door behind h...
Person: I -[HAS_IN_POSSESSION]-> Object: stone	thought of the agonies I had gone through i...

2. Graph Retrieval strategies – Generation

Lastly: generate the answer!

Langchain: wrap the prompt in a **VectorStoreRetriever**
Oracle DB: **DBMS_CLOUD_AI.GENERATE**

```
DBMS_CLOUD_AI.GENERATE(prompt => '
   Use the following pieces of retrieved context to answer the question:

   Person: Ryder –[COMMITTED_CRIME]-> Crime: theft

   Source: very scrupulous in the means you used. It seems to me, Ryder, that there is
   the making of a very pretty villain in you. You knew …
   ---
   Question: Who stole the jewel?
', action => 'chat')
```

Correct context matters!

Answer: **Ryder stole the jewel.** He committed the crime with the help of his confederate Cusack.

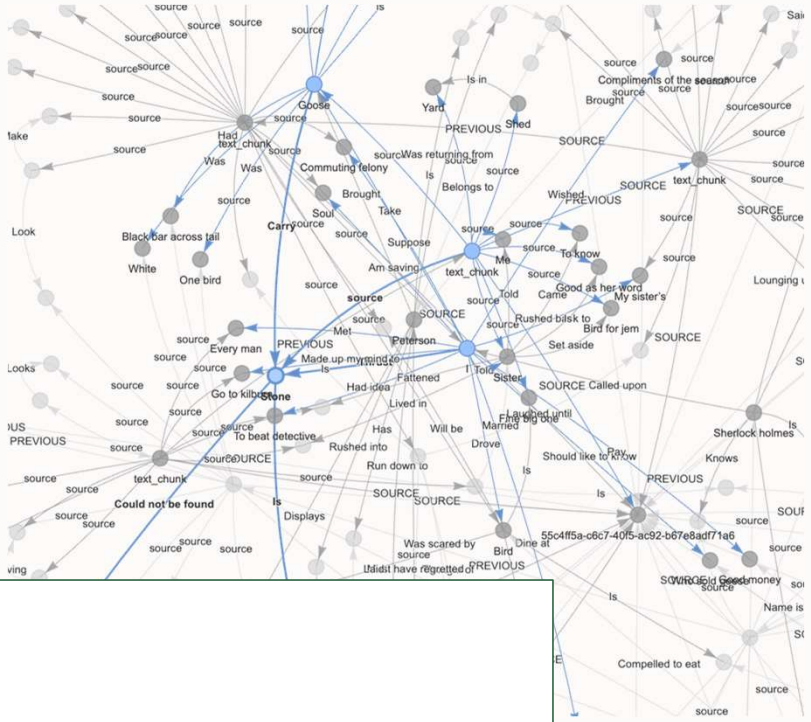
Going beyond triples

Until now: information extraction to improve retrieval quality

- Many other techniques trying to do this
  - reranking, in-context embeddings…

Is there more to Graphs?

- We worked with a Graph, but didn't really treat is as such
  - Storing the data as **(source) –[relation]→ (target)**
  - How about going over multiple hops?




```
CREATE PROPERTY GRAPH RAG_GRAPH
VERTEX TABLES (
   GRAPH_ENTITIES LABEL ENTITY
) EDGE TABLES (
   GRAPH_RELATIONS          KEY(ID)
            SOURCE          KEY head REFERENCES GRAPH_ENTITIES (entity_name)
            DESTINATION     KEY tail REFERENCES GRAPH_ENTITIES (entity_name)
            LABEL           RELATION PROPERTIES (RELATION, EMBEDDING))
)
```

Querying a Property Graph

Find out how **Ryder** is connected to the **stone**

- Examine Relationships up to 3 hops

Reading the output

1. Ryder asks Holmes
   Holmes Interrogates Ryder
   Ryder had a stone

2. Ryder had a stone
   Stone was in a goose!
   Goose had a stone

… more complex relationships beyond the scope of the example …

```
SELECT * FROM GRAPH_TABLE (PG_RAG
   MATCH (A IS ENTITY) ((v1) -[E1]-> (v2)){1,3} (B IS ENTITY)
   WHERE A.ENTITY_NAME = 'Ryder' AND
         B.ENTITY_NAME = 'stone'
   COLUMNS(
      LISTAGG(V1.ENTITY_NAME, ', ')    AS A_NAME,
      LISTAGG(E1.RELATION, ',')        AS E_TYPE,
      LISTAGG(V2.ENTITY_NAME, ',')     AS B_NAME
   )
)
```

A_NAME	E_TYPE	B_NAME
Ryder	HAD	stone
Ryder, Holmes, Ryder	ASKS,INVESTIGATES,HAD	Holmes,Ryder,stone
Ryder, Holmes, Ryder	ASKS,INTERROGATES,HAD	Holmes,Ryder,stone
Ryder, Holmes, Ryder	ASKS,SAYS_TO,HAD	Holmes,Ryder,stone
Ryder, stone, goose	HAD,WAS_IN,HAD	stone,goose,stone
Ryder, stone, goose	HAD,CAME_FROM,HAD	stone,goose,stone
Ryder, Holmes	ASKS,HAS_ITEM	Holmes,stone

RAG with Property Graphs: Outlook

Ongoing exploration

Takeaways
- Tools are there
- Need to explore more complex use-cases

With Graphs we can connect
- Extracted Knowledge with Relational data
  - Traverse through numerical data connected to embeddable entities
- Combine Semantic and Traditional search
  - Use hybrid queries to match patterns in the Graph
  - Retrieve complete information based on embedded entities

Medical domain
- *What are the symptoms of diseases commonly treated with the given medication?*

Financial transactions
- *Find circular patterns in transactions between entities with similar names*

Tools: Graph RAG support with Oracle DB

Step	Oracle in-DB support	Third-party support
Document Loading, Chunking, and Embedding	DBMS_CLOUD_AI (high-level API) • **CREATE_VECTOR_INDEX** DBMS_VECTOR_CHAIN • **UTL_TO_TEXT** • **UTL_TO_CHUNKS** • **UTL_TO_EMBEDDING** (ONNX or OCI Gen AI) OML4SQL (ONNX) • **VECTOR_EMBEDDING**	Langchain • OracleDocLoader • OracleDocReader • OracleTextSplitter • OracleEmbeddings • OCIGenAIEmbeddings Many other Langchain and LlamaIndex methods
Entity and Relationship extraction	**DBMS_CLOUD_AI** (manual work) • **GENERATE** with **chat** action and **JSON_TABLE** for output parsing	Langchain **LLMGraphTransformer** LlamaIndex **PropertyGraphIndex**
Storage, Retrieval, and Generation	**Property Graphs queries** **DBMS_CLOUD_AI** (manual work) • **GENERATE** with **chat** action • Manual context creation and templating	Langchain • ChatOCIGenAI • Oracle Vector Store • manual work for the retriever

Open questions and challenges

Takeaways

- You can do Graph RAG in and with Oracle DB
- Explicitly extracting information from unstructured text can improve retrieval quality
- Graphs can explicitly capture relationships in unstructured data

Open challenges

- What are the best retrieval strategies?
  - For unstructured data
  - For Hybrid approaches between unstructured and relational data
- How to approach entity resolution?
- Manual effort required to implement Graph RAG in the DB
- Evaluation and enterprise use-cases

Thank you


ORACLE