# GraphRAG with Neo4j and LLMs

Prerequisite Concepts and Technologies

# Section 1

Knowledge Graphs

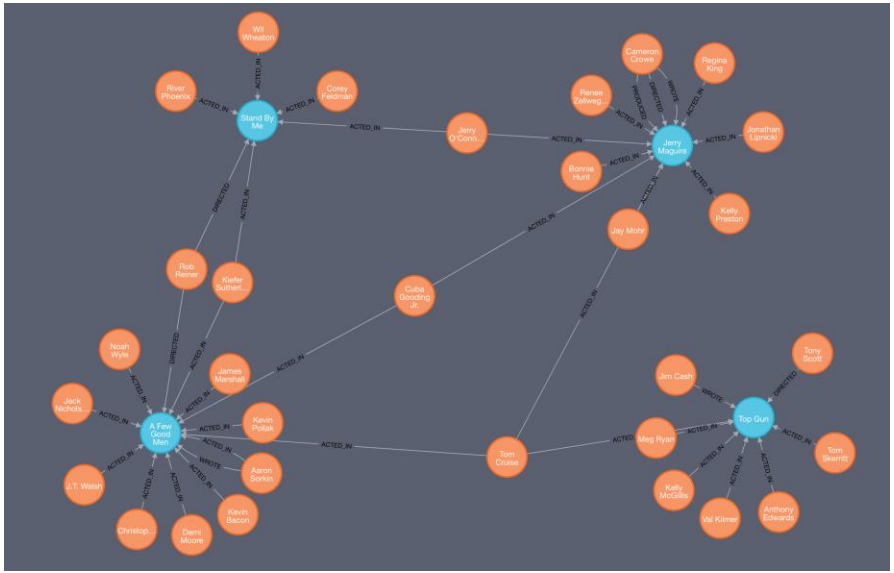
# What is a Knowledge Graph?

## Definition

A knowledge graph is a structured representation of information that captures entities and their relationships in a network format.

## Key Characteristics

- Entities represented as nodes
- Relationships as edges
- Properties on both nodes and edges
- Semantic meaning encoded in structure



# Knowledge Graph Components

## Nodes (Entities)

Represent real-world objects

Examples: Person, Movie, Company

## Edges (Relationships)

Connect entities meaningfully

Examples: ACTED_IN, WORKS_FOR

## Properties (Attributes)

Store additional information

Examples: name, age, date

**Example:** (Actor)-[ACTED_IN {role: "Neo"}]->(Movie)

# Graph vs. Relational Databases

## Relational (SQL)

- Tables with rows and columns
- Joins for relationships
- Schema-rigid structure
- Optimized for transactions
- Slower for deep relationships

## Graph (Neo4j)

- Nodes and relationships
- Direct connections (no joins)
- Flexible schema
- Optimized for connections
- Fast multi-hop queries

# Knowledge Graph Use Cases

### 🔍 Search & Discovery

Google, Amazon product graphs

### 💊 Healthcare

Drug interactions, patient records

### 🛡️ Fraud Detection

Financial transaction networks

### 🤖 AI & ML

Context for LLMs, recommendations

### 🕸️ Social Networks

Friend connections, content graphs

### 🏢 Enterprise

Data lineage, organizational charts

# Section 2

Neo4j Database

# Introduction to Neo4j

Neo4j is the world's leading **graph database** platform with native graph storage and processing.

## Key Features

- ACID compliant transactions
- Cypher query language
- Scalable to billions of nodes
- Graph algorithms library
- Cloud-native (AuraDB)

Graph Native

# Graph Database Concepts

## Index-Free Adjacency

Each node directly references its neighbors, eliminating the need for index lookups during traversal

### Labels

Categorize nodes (e.g., :Person, :Movie)

### Relationship Types

Define connection types (e.g., ACTED_IN)

**Each node maintains a constant-time pointer list to adjacent edges and connected nodes. Therefore, traversal from one node to another is O(1) (constant time) per relationship.**

# Neo4j Architecture Layers

### Cypher Query Language

Declarative graph query language

### Graph Engine

Query execution and transactions

### Page Cache

In-memory caching

### Native Graph Storage

Persistent storage

# Neo4j Architecture Layers

| Layer | Role | Analogy |
| --- | --- | --- |
| **Cypher Query Language** | User-facing interface | How you speak to the graph |
| **Graph Engine** | Executes Cypher, manages transactions | The CPU |
| **Page Cache** | Keeps hot data in memory | The RAM |
| **Native Graph Storage** | Stores the full graph on disk | The Hard Drive |

# Neo4j AuraDB Cloud

Fully managed cloud database service for Neo4j - perfect for labs and production

### Cloud Native
No installation required

### Secure
Encrypted connections

### Fast
SSD-backed storage

## Free Tier Available!

Perfect for learning and development

# Section 3

Cypher Query Language

# Cypher Query Language

Cypher is Neo4j's **declarative graph query language** - like SQL for graphs but with ASCII-art syntax

## Why Cypher?

- Intuitive visual syntax
- Pattern matching
- Expressive and readable
- Optimized for graphs

```
// Example Pattern
(actor)-[:ACTED_IN]->(movie)

Nodes in parentheses
Relationships in brackets
```

# Basic Cypher Syntax

```
// CREATE: Add nodes and relationships
CREATE (n:Person {name: 'Alice'})
```

```
// MATCH: Find patterns in the graph
MATCH (p:Person) RETURN p.name
```

```
// WHERE: Filter results
MATCH (p:Person) WHERE p.age > 21
```

```
// WITH: Chain operations
MATCH (p) WITH p.name AS name RETURN name
```

# Common Cypher Patterns

Find all actors in a movie

```
MATCH (a:Actor)-[:ACTED_IN]->(m:Movie {title: 'Inception'})
RETURN a.name
```

Find shortest path between nodes

```
MATCH p=shortestPath((a:Person)-[*]-(b:Person))
RETURN p
```

Aggregate and count

```
MATCH (d:Director)-[:DIRECTED]->(m:Movie)
RETURN d.name, count(m) AS movies
```

# Section 4

Large Language Models

# Large Language Models (LLMs)

LLMs are neural networks trained on vast text corpora to understand and generate human language

**Scale**
Billions of parameters

**Training**
Massive text datasets

**Versatility**
Multi-task capable

**Examples:** GPT-4, Llama, Mistral, Claude, Gemini

# Transformer Architecture

The transformer architecture revolutionized NLP with its attention mechanism

## Key Components

- Self-attention layers
- Positional encodings
- Feed-forward networks
- Layer normalization

### Attention Is All You Need

Vaswani et al., 2017

**Benefits:**

- Parallel processing
- Long-range dependencies
- Scalable architecture

# Hugging Face Ecosystem

The leading platform for machine learning models and datasets

### 🤗 Model Hub

500,000+ pre-trained models

### 📚 Transformers Library

Easy model deployment

### 💾 Datasets

Curated dataset library

### 🚀 Inference API

Hosted model endpoints

# Popular Open-Source LLMs

**Mistral 7B**
Powerful 7B parameter model

**Llama 2**
Meta's open model family

**Gemma**
Google's lightweight models

**Phi-3**
Microsoft's efficient models

# LLM Capabilities

### Text Generation

Creative writing, code

### Question Answering

Information retrieval

### Summarization

Condensing information

### Translation

Multilingual support

### Classification

Sentiment, categorization

### Reasoning

Logic and inference

# Section 5

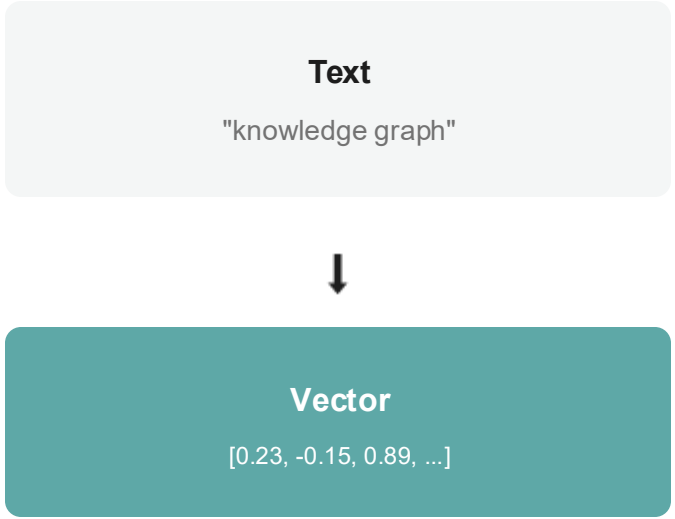
Vector Embeddings

# What are Embeddings?

Embeddings are **dense vector representations** of text that capture semantic meaning in continuous space

## Characteristics

- Fixed-length vectors (e.g., 384, 768, 1536 dimensions)
- Semantically similar texts have similar vectors
- Learned from large text corpora
- Enable mathematical operations on meaning



# Vector Space Properties

## Proximity = Similarity

Vectors close together in space represent similar concepts

### Example Cluster

"king" ≈ "queen" ≈ "monarch"

"dog" ≈ "puppy" ≈ "canine"

### Vector Operations

king - man + woman ≈ queen

Arithmetic on meaning!

# Semantic Similarity

Measure similarity between vectors using distance metrics

### Cosine Similarity

Measures angle between vectors (-1 to 1)

Most common for text embeddings

### Euclidean Distance

Straight-line distance in vector space

**Higher similarity = More relevant results**

# Popular Embedding Models

### all-MiniLM-L6-v2

Fast and lightweight, great for prototypes

### text-embedding-ada-002

OpenAI's production embedding model

### BGE-large-en

State-of-the-art open-source model

# Section 6

Retrieval-Augmented Generation

# Retrieval-Augmented Generation (RAG)

RAG enhances LLM outputs by retrieving relevant information before generation

### 1. Retrieve

Find relevant documents or data

### 2. Augment

Add retrieved context to prompt

### 3. Generate

LLM creates informed response

# RAG System Architecture

User Query

Embedding Generation

Vector Search

Context + Prompt

LLM Response

Flow

# Why Use RAG?

### ✅ Reduces Hallucinations
Grounds responses in facts

### 📚 Up-to-date Information
Access latest data

### 🎯 Domain-Specific
Custom knowledge bases

### 🔍 Source Attribution
Traceable answers

**RAG > Fine-tuning for many use cases**

# Vector Search in RAG

## How It Works

- Convert query to embedding
- Compare with stored vectors
- Return top-k similar items
- Use as context for LLM

**Advantage:** Captures semantic meaning beyond keywords

## Vector Databases

- Neo4j (with vector index)
- Pinecone
- Weaviate
- Qdrant
- Milvus

# Classical RAG Limitations

Traditional RAG systems face several challenges with complex queries

### ❌ Isolated Information

Documents retrieved independently without understanding connections

### ❌ No Relationship Context

Cannot traverse entity relationships or multi-hop reasoning

### ❌ Limited Semantic Understanding

Struggles with complex queries requiring domain knowledge

# Full-Text Search

Traditional keyword-based search using lexical matching

## Best For

- Exact phrase matching
- Names and identifiers
- Dates and numbers
- Domain-specific terms

## Limitations

- No semantic understanding
- Misses synonyms
- Word order matters
- Less flexible

Complements vector search for comprehensive retrieval

# Section 7

GraphRAG

# What is GraphRAG?

## GraphRAG = Knowledge Graphs + Retrieval-Augmented Generation

Combines the structured relationships of knowledge graphs with the semantic understanding of LLMs for more accurate and contextual retrieval

**Key Innovation**
Leverages both graph structure and vector embeddings to retrieve richer, more connected information

# GraphRAG: Deep Dive

## How It Works

- User query converted to embedding
- Vector search finds relevant nodes
- Graph traversal explores relationships
- Full-text search captures exact terms
- Context assembled from graph structure
- LLM generates informed response

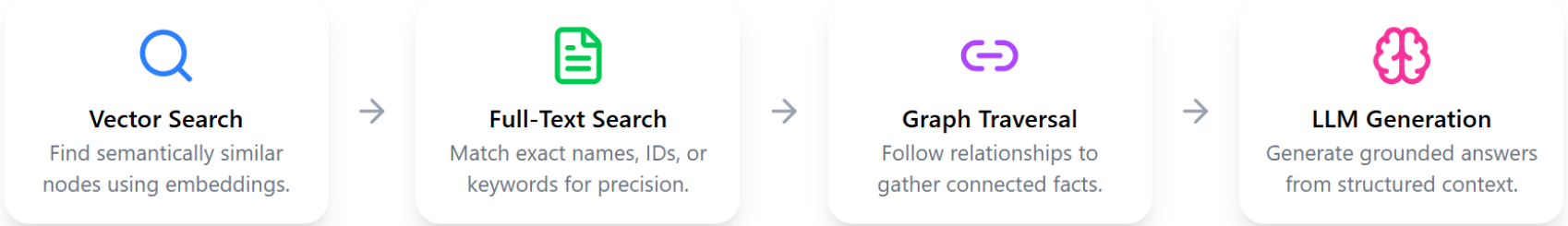
## Key Advantages

- Preserves entity relationships
- Enables multi-hop queries
- Combines multiple retrieval methods
- Provides explainable results
- Better contextual understanding

# GraphRAG: Deep Dive

## GraphRAG Pipeline

Combining **vector search**, **full-text search**, and **graph traversal** to retrieve rich context for LLM-based reasoning.




**GraphRAG = Vector Search + Full-Text Search + Graph Traversal → Context → LLM → Explainable Answer**

# GraphRAG vs. Traditional RAG

## Traditional RAG

- Document-based retrieval
- Flat vector search
- Limited relationship context
- Isolated information

## GraphRAG

- Entity & relationship aware
- Graph traversal + vectors
- Rich contextual connections
- Multi-hop reasoning

# Hybrid Retrieval Strategy

Combining multiple search methods for optimal results

### Vector Search

Semantic similarity

### Full-Text Search

Keyword matching

### Graph Traversal

Relationship exploration

# Hybrid Retrieval: The Complete Picture

## Hybrid Retrieval in GraphRAG

Example query: **"What proteins interact with drugs that treat hypertension?"**


**Hybrid Retrieval = Vector Search + Full-Text Search + Graph Traversal → LLM → Explainable, Connected Answer**

# Benefits of GraphRAG

### 🎯 Better Accuracy
More relevant context through relationships

### 🔗 Rich Context
Connected information, not isolated facts

### 🧠 Complex Queries
Handle multi-hop reasoning

### 📊 Explainable
Traceable through graph structure

**The future of intelligent QA systems**

# Section 09

Putting It All Together

# GraphRAG Best Practices

**✓ Design Good Schema**
Clear labels, meaningful relationships

**✓ Create Indexes**
Vector and full-text for performance

**✓ Use Quality Embeddings**
Choose appropriate model size

**✓ Optimize Cypher**
Profile queries, use LIMIT wisely

**✓ Test Retrievers**
Evaluate precision and recall

**✓ Monitor Performance**
Track latency and accuracy

# Common Pitfalls to Avoid

### ⚠️ Missing Indexes
Slow queries without proper vector/fulltext indexes

### ⚠️ Over-retrieval
Fetching too much context overwhelms the LLM

### ⚠️ Poor Graph Design
Unclear relationships make traversal ineffective

### ⚠️ Ignoring Graph Structure
Using only vector search defeats the purpose of GraphRAG

# GraphRAG Real-World Applications

### Healthcare
Drug interactions, medical knowledge graphs, patient care paths

### Finance
Risk analysis, fraud detection, entity relationships

### Research
Scientific literature analysis, citation networks

### Enterprise
Knowledge management, organizational intelligence

### E-commerce
Product recommendations with relationship context

### Cybersecurity
Threat intelligence, attack pattern analysis

# GraphRAG Real-World Applications

## Empathize

### Who are your users?

- Students
- Doctors
- Researchers
- Travelers
- Entrepreneurs

### What do they need?

- Quick answers
- Connections between data
- Trusted sources

### Pain points

- Too much unstructured info
- Hard to find related facts
- Time-consuming search

## Define

### Problem Statement

- "Users need a way to ___ because ___"
- Example: "Researchers need a way to link articles and datasets because information is siloed."

### Knowledge Graph Domain

- Healthcare
- Education
- Tourism
- Law
- Mobility

### Entities & Relationships

- Drug–Disease
- Student–Course–Topic
- Place–Review–Activity

## Ideate

### Possible Solutions

- Chatbot interface
- Recommendation panel
- Interactive graph viewer

### Retrieval Strategies

- Vector search for semantic similarity
- Full-text for exact terms
- Graph traversal for multi-hop context

### LLM Outputs

- Summaries
- Explanations
- Recommendations
- Insights with citations

# Section 8

LangGraph Framework

# What is LangGraph?

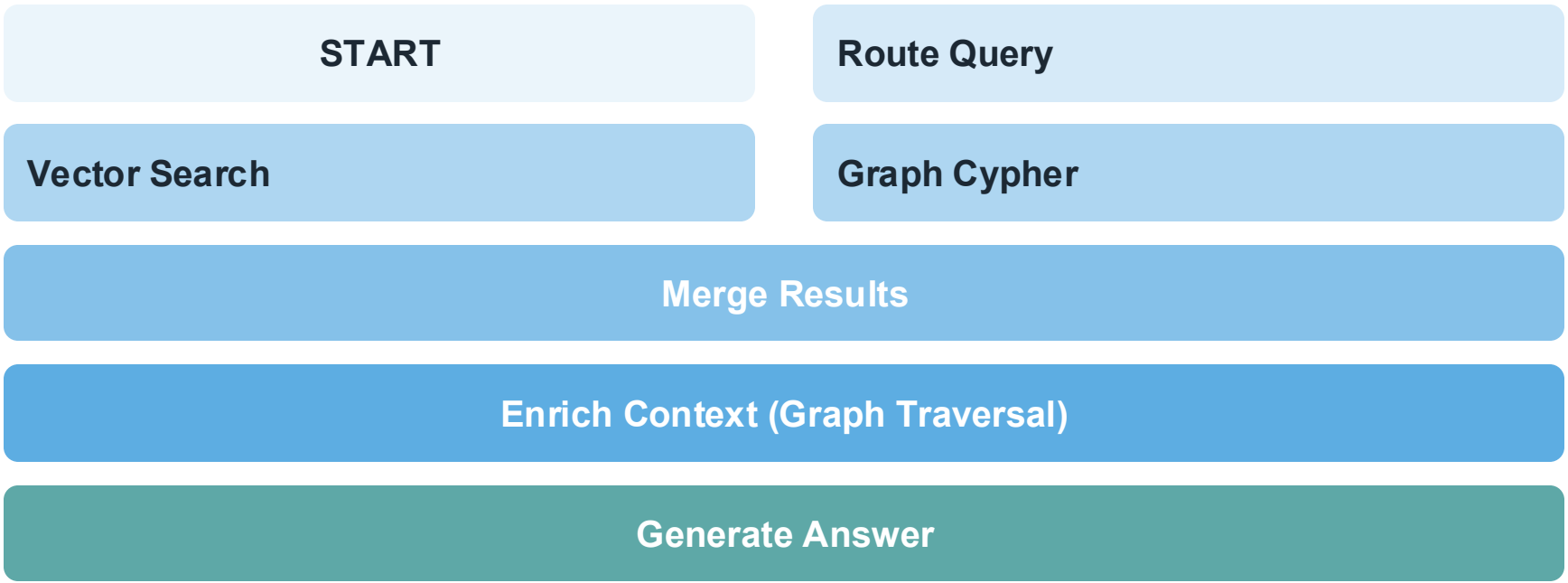
LangGraph extends LangChain with stateful, cyclical workflows for building complex agent systems

### Cycles & Loops
Support for iterative and conditional workflows

### State Management
Persistent state across workflow steps

### When to Use LangGraph
- Multi-step reasoning with decision points
- Agent-based systems that need memory
- Complex workflows with conditional logic
- Systems requiring human-in-the-loop

# LangGraph: GraphRAG Workflow




*Conditional routing and loops enable complex reasoning*