✦ PLAYBOOK · RAG SYSTEMS

Knowledge Graphs For RAG

When a graph improves retrieval, how to extract entities and relations, design a schema that lasts, and combine graph and vector search in production.

FORMAT	LEVEL	READ TIME	UPDATED
Playbook	Advanced	15 min	Oct 2026

OVERVIEW

What this playbook covers

Vector search finds passages that look like the question. It struggles when the answer depends on connections: who reports to whom, which component depends on which supplier, what changed across a series of documents. Knowledge graphs make those connections explicit. This playbook covers extracting entities and relations with LLMs, designing a schema you can maintain, GraphRAG patterns, hybrid graph-plus-vector retrieval, keeping the graph current, and the many cases where a graph is not worth the effort.

KEY TAKEAWAYS

- Graphs help with multi-hop, aggregate and relationship questions; they add little to simple lookup questions.
- Start from the questions users ask and design the smallest schema that answers them.
- Entity resolution — deciding that two mentions are the same thing — is where graph quality is won or lost.
- Hybrid retrieval uses vectors to find entry points and the graph to expand along relationships.
- Every node and edge should trace back to the source text that produced it.

WHO IT'S FOR

AI engineers building advanced RAG

Data and knowledge engineers

Architects evaluating GraphRAG

Product owners of research and analysis tools

HOW TO USE IT

Read section 06 first if you are unsure a graph is needed. Otherwise sections 01–02 define the build, 03–04 the retrieval design, and 05 the operating model.

CONTENTS

01 Design the schema from the questions
02 Entity and relation extraction
03 GraphRAG patterns
04 Hybrid graph and vector retrieval
05 Keeping the graph current
06 When not to bother

01 / SCHEMA

Design the schema from the questions

An ontology designed in the abstract grows until nobody can populate it. Collect the questions users need answered and work backwards to the entity and relation types they require.

01 Collect target questions

Gather real questions, especially those current RAG answers badly — multi-hop, comparative or “all X that…”.

02 Mark the entities and relations

For each question, underline the things and the connections it depends on.

03 Define a small set of types

Name entity and relation types with clear definitions and examples; resist adding types no question needs.

04 Decide on properties

Choose attributes such as dates, amounts and status that questions filter or compare on.

05 Write it down as a contract

The schema is used by extraction prompts, validators and queries alike, so version it.

Example: supplier risk schema

ELEMENT	TYPE	EXAMPLE PROPERTIES
Company	Entity	Name, country, identifiers
Component	Entity	Part number, category
SUPPLIES	Relation · Company → Component	Contract start, end, volume
LOCATED_IN	Relation · Company → Region	Site type
MENTIONED_IN	Relation · any → Document	Page, span, extraction date

02 / EXTRACTION

Entity and relation extraction

LLMs make extraction from unstructured text practical, but raw extractions are noisy. Treat extraction as a pipeline with validation, not a single prompt.

01 Chunk source documents → 02 Extract to schema (JSON) → 03 Validate types & fields → 04 Resolve entities → 05 Write nodes & edges with provenance

01

Constrain to the schema

Give the extractor the allowed entity and relation types and a JSON schema. Reject anything outside it.

02

Resolve entities deliberately

Combine exact identifiers, normalized names, embeddings and rules to merge mentions. Keep merges reversible.

03

Keep provenance on everything

Each node and edge stores the document, chunk and text span it came from, so answers can cite it.

04

Use structured sources first

Where a database or system of record already holds a relationship, import it rather than extracting it from text.

05

Score confidence

Store an extraction confidence and filter or review low-confidence edges before they reach answers.

06

Measure on a labeled sample

Hand-label a sample of documents and measure extraction precision and recall before scaling up.

FIELD NOTE

Entity resolution is the hard part

“Acme Ltd”, “ACME” and “Acme Holdings” may or may not be the same company. Wrong merges create false connections; missed merges hide real ones. Budget real time for this step.

✦ 03 / PATTERNS

GraphRAG patterns

"GraphRAG" covers several distinct approaches. Pick by the kind of question you need to answer.

LOCAL

Entity neighborhood

Find the entities in the question, pull their nearby nodes and source chunks, and answer from that subgraph.

GLOBAL

Community summaries

Cluster the graph into communities, pre-summarize each, and answer broad "what are the themes" questions from summaries.

QUERY

Text-to-graph-query

Translate the question into a graph query such as Cypher for precise, aggregate or filtered answers.

PATH

Multi-hop traversal

Follow relations across several hops to connect facts that live in different documents.

HYBRID

Graph-guided retrieval

Use graph neighbors to expand or re-rank the chunks vector search returns.

STRUCTURED

Graph as metadata

Use entity links only as filters on ordinary chunk retrieval — the lightest-weight option.

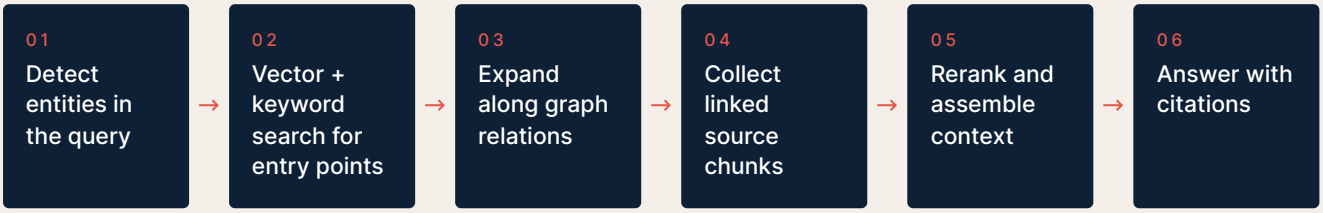
VECTOR-ONLY RAG ANSWERS WELL
"What does the policy say about X?"
Single-document lookups
Paraphrased questions
Questions about one entity

GRAPH-AUGMENTED RAG ANSWERS BETTER
"Which suppliers of X are in region Y?"
Questions spanning many documents
Counting, grouping and comparison
"What connects A and B?"

04 / RETRIEVAL

Hybrid graph and vector retrieval

In practice the strongest systems use both: vectors to find where to start, the graph to follow what is connected, and the original text to ground the answer.



01 **Link query to entities**

Match names in the question to graph nodes using the same resolution logic as ingestion.

02 **Bound the expansion**

Limit hops and fan-out per hop; unbounded traversal pulls in noise and blows the context budget.

03 **Return text, not just triples**

Pass the source passages behind each edge to the model so the answer is grounded in real wording.

04 **Apply permissions to the graph**

Edges inherit the access rights of their source documents; a user should not see a relation derived from a document they cannot read.

05 **Route by question type**

Send lookups to plain retrieval and relationship or aggregate questions to the graph path.

05 / OPERATIONS

Keeping the graph current

A graph built once and never updated becomes a confident source of stale answers. Maintenance is part of the design.

UPDATES

- [ ] Incremental extraction from document change feeds

- [ ] Edges removed when their only source document is deleted
- [ ] Temporal properties on relations that change over time

QUALITY

- [ ] Periodic review of low-confidence and high-degree nodes
- [ ] Merge and split tools for entity resolution errors
- [ ] Extraction precision re-measured after prompt or model changes

SCHEMA

- [ ] Schema versioned with migration scripts
- [ ] New types added only for questions that need them
- [ ] Unused types retired

EVALUATION

- [ ] Golden questions that need multi-hop answers
- [ ] Comparison against vector-only baseline on every change
- [ ] Citations checked back to source spans

✦ 06 / JUDGMENT

When not to bother

Graphs add extraction cost, a second store, and an ongoing maintenance load. Many RAG systems are better served by fixing chunking, hybrid search and reranking first.

SITUATION	BETTER FIRST STEP
Questions are mostly single-document lookups	Improve chunking, hybrid search and reranking
Relationships already live in a database	Query the database directly as a tool
Corpus is small enough to fit in context	Long-context prompting with good structure

SITUATION	BETTER FIRST STEP
Nobody can own entity resolution and schema	Use metadata filters instead of a full graph
Content changes faster than extraction can keep up	Retrieval over fresh text, with graph only for stable entities

> Build the graph for the questions that need it — and measure that it beats the simpler system before you keep it.
>
> SPLENDENSLABS ENGINEERING PRINCIPLE

GET IN TOUCH

Let's Talk

We build hybrid graph and vector retrieval for teams whose questions span documents, systems and relationships — and tell you honestly when a graph is not needed.

- Question analysis and graph-fit assessment
- Schema and extraction pipeline design
- Hybrid GraphRAG retrieval build
- Evaluation against your current RAG baseline

BOOK AN AI STRATEGY SESSION →

EMAIL
hello@splendenslabs.com

WEB
splendenslabs.com

OFFICE
Sona Incubation Foundation, Salem