1. **Legal schema & exchange governance (beyond modeling).**

Adopt **Akoma Ntoso** for document structure (articles/clauses/amendments) and **ELI** for legislation metadata/IDs, even if your operational store is a property graph. This future-proofs interop and publication.

1. **Provenance & chain-of-custody (first-class, not optional).**

Decide early how you’ll record *who/when/how* each edge/fact was derived. Mirror **W3C PROV-O** terms (agent/activity/entity) and keep them on every statement; you’ll need this for audit trails and courtroom defensibility.

1. **Bi-temporal & temporal graph design (formal patterns, not just timestamps).**

Lock down **valid time vs transaction time** semantics and “as-of” query patterns; study bitemporal references and temporal PG research so your snapshot/diff queries scale. (Neo4j has versioning patterns; Aion shows efficient temporal LPG storage.)

1. **Normative conflict & rule reasoning (beyond RAG).**

Plan a track for *rule-aware* checks: **LegalRuleML** (OASIS) for machine-interpretable norms, **LKIF-Core** as a legal ontology backbone, and recent **deontic logic** work for conflict detection/“ought” reasoning—useful when surfacing contradictions or drafting harmonized language. ([ceur-ws.org](https://ceur-ws.org/Vol-321/paper3.pdf?utm_source=chatgpt.com), [Enciclopédia de Filosofia de Stanford](https://plato.stanford.edu/entries/logic-deontic/?utm_source=chatgpt.com))

1. **Schema/shape validation (quality gates).**

Even if you stay property-graph, learn from **W3C SHACL** (RDF constraint language) to express structural/temporal/provenance constraints; translate those constraints into Cypher/Gremlin checks or run SHACL on an export. It’s the cleanest way to keep the KG consistent. ([W3C](https://www.w3.org/TR/shacl/?utm_source=chatgpt.com))

1. **Arabic-first NLP & bilingual normalization.**

Treat **Arabic OCR + morphology** as a dedicated stage: segmentation/lemmatization/NER with **CAMeL Tools**; consider **Farasa** for fast segmentation. These choices materially affect search, entity linking, and citation extraction. ([aclanthology.org](https://aclanthology.org/2020.lrec-1.868.pdf?utm_source=chatgpt.com), [lrec-conf.org](https://www.lrec-conf.org/proceedings/lrec2016/pdf/164_Paper.pdf?utm_source=chatgpt.com))

1. **Observability for GenAI/GraphRAG (not just logs).**

Instrument with **OpenTelemetry GenAI semantic conventions** (spans for LLM calls, agent tool calls, metrics for token/latency) and add a pipeline-aware evaluator like **Arize Phoenix** or **TruLens** (RAG Triad: context relevance, groundedness, answer relevance). This de-risks production.

1. **Safety & red-teaming specific to LLMs.**

Map your threat model to **OWASP Top 10 for LLMs** (prompt injection, insecure output handling, data poisoning, etc.), then wire preventative controls/safeguards (e.g., guarded tool use, sanitizers) and use Azure’s groundedness checks as a final gate before display.

1. **Evaluation protocol for legal QA (acceptance criteria).**

Beyond offline IR metrics, standardize **reference-free evaluations** (RAG Triad) and a *jurist review* loop for tricky questions. Keep a gold set of “as-of” temporal queries and contradiction cases; automate triad scoring in CI.

1. **Global/community summarization design (GraphRAG-specific).**

If you’ll use **Global** or **DRIFT**, plan the community-detection & summarization cadence and storage (versions, “summary\_as\_of”), following the **Microsoft GraphRAG** docs. This is a distinct indexing pipeline, not just retrieval.

1. **Data acquisition & update operations (regulatory watch).**

Treat source crawling/OCR/structuring as a product: gazette monitors, schedule, retries, change detection, and fact retraction propagation. Microsoft’s GraphRAG materials emphasize *regular* rebuilds of graphs + summaries to keep “global” fresh.

1. **Visualization & audit UX plan (who needs what view).**

Decide early: which users need **Bloom-style** graph paths vs. tabular evidence packets vs. timeline diff views? Neo4j’s Bloom is a great audit affordance; if your core is Azure PaaS, mirror to Neo4j read-only for investigations. (Neo4j also documents GraphRAG patterns & tooling you can leverage even if it’s a mirror.)

**Where I’d personally research one level deeper next**

* **Normative reasoning & conflicts** (LegalRuleML + deontic logic options and how to *operationalize* them in your stack). ([proceedings.kr.org](https://proceedings.kr.org/2024/7/kr2024-0007-arieli-et-al.pdf?utm_source=chatgpt.com))
* **Temporal PG at scale** (point-in-time vs. diff queries, retention, and cost; Aion & Neo4j versioning patterns).
* **Arabic NLP impact on retrieval** (tokenization/diacritics/lemmatization tradeoffs with CAMeL/Farasa and their effect on hybrid search). ([aclanthology.org](https://aclanthology.org/2020.lrec-1.868.pdf?utm_source=chatgpt.com), [lrec-conf.org](https://www.lrec-conf.org/proceedings/lrec2016/pdf/164_Paper.pdf?utm_source=chatgpt.com))
* **Production observability & eval** (OpenTelemetry GenAI + Phoenix/TruLens baked into CI/CD and SRE dashboards).