

Governed Enterprise AI Memory Beyond RAG: From Vector Retrieval to Permissioned Knowledge Graphs

Zhirong (Mark) Huang
ORCID: [0009-0007-4326-2042](https://orcid.org/0009-0007-4326-2042)

2026-06-02

Abstract

Enterprises are adopting multiple AI tools at the same time: chat assistants, coding agents, office copilots, search systems, and workflow automations. These systems can produce useful work when the necessary background information is available in the prompt, context window, connector, or retrieval layer. The failure mode is that organizational background knowledge is often missing, stale, fragmented across teams, or inaccessible to the AI client performing the task. In that case, the model may generate plausible output from pretrained knowledge and local prompt clues rather than from the current state of the organization.

Retrieval-augmented generation (RAG) is the common response to this problem and has become a central technique for grounding large language models in external knowledge [2]. RAG helps, but vector retrieval alone does not inherently define which fact is current, which source is authoritative, which older decision has been superseded, or which actor is allowed to read or mutate knowledge. This article proposes governed enterprise AI memory as a layer above ordinary retrieval: a permissioned, provenance-preserving knowledge graph that can connect multiple AI clients while maintaining team isolation, read/write scopes, evidence lineage, conflict handling, and fact supersession.

The paper uses Dense-Mem as a reference implementation example, not as the sole subject of the contribution. It also defines a benchmark plan for testing long-context prompting, vector RAG, graph retrieval, and governed memory on current facts, stale facts, conflicts, missing evidence, cross-team isolation, readonly access, and multi-client continuity. The package includes an initial 100-scenario synthetic comparison between RAG over source documents and a curated knowledge graph baseline built from the same scenario information. This revision also describes the reference implementation’s process path from source fragments through typed claims, verification, promotion gates, tiered recall, and scoped memory export/import for portable agent knowledge and history.

1. Introduction

Large language models are useful because they can condition generation on a prompt and its surrounding context. The Transformer architecture introduced a sequence-transduction model built around self-attention that underlies many modern LLM systems [1], and retrieval-augmented

generation later showed a practical way to combine parametric model behavior with external documents [2]. The enterprise problem is not only whether a model is capable of reasoning over text. The problem is whether the correct organizational background information is available, current, permitted, and traceable when the model is asked to answer.

Modern companies rarely use a single AI surface. A developer may use Claude Code or another coding agent inside a repository. A product manager may use ChatGPT or a workplace copilot to summarize customer requests. A support team may use an AI workflow that reads tickets. A business stakeholder may use a chat assistant to draft a roadmap update. Each surface can be valuable, but each surface often has a partial view of the organization.

The result is a fragmented-context failure mode:

- Business context exists but is not visible to the developer’s coding agent.
- Engineering constraints exist but are not visible to a business user’s AI assistant.
- A prior decision has been superseded, but an older document remains easier to retrieve.
- A model can retrieve semantically similar text, but not determine whether that text is current, authoritative, or allowed for the current actor.
- Automation can act with confidence while using incomplete or stale background information.

Industry reports indicate that organizational AI adoption is widespread, while scaled impact remains harder than adoption alone. McKinsey’s 2025 global survey reported broad AI use, early agentic AI experimentation, and a gap between deployment and enterprise-level financial impact [16]. Stanford HAI’s 2026 AI Index reported that organizational AI adoption continued to rise in 2025 [17]. These adoption signals make the context problem more important: the more AI tools an enterprise deploys, the more important shared, governed organizational memory becomes.

1.1 Evidence Map And Claim Boundary

The evidence base is uneven, so the article separates motivation, prior art, and implementation evidence instead of treating them as one proof.

Article claim	Current support	Strength	Boundary
Enterprise AI adoption is broad, while scaled impact remains uneven.	Stanford reports 88% organizational AI adoption in 2025, and McKinsey reports 23% of respondents scaling agentic AI systems and 39% reporting enterprise-level EBIT impact [17, 16].	Public survey evidence.	These surveys motivate the problem; they do not prove that governed memory is the only solution.
Larger context windows do not remove the need for explicit memory governance.	Lost-in-the-Middle, RULER, and Context Rot report reliability limits for long-context model use [3, 5, 15].	Academic and technical evaluation evidence.	These studies support caution about long context; they do not show that every model fails on every long-context task.

Enterprise RAG evaluation is moving toward company-internal workflows.	EKRAG, WixQA, and EnterpriseRAG-Bench evaluate enterprise or support-style retrieval and question answering [8, 9, 14].	Benchmark and dataset evidence.	These benchmarks cover retrieval and answering more than write governance, supersession, team isolation, and multi-client memory continuity.
Graph-backed or temporal memory is active prior art.	GraphRAG, knowledge graph-guided RAG, Zep, Governed Memory, Context Objects, and Stateless Decision Memory all overlap with this design space [4, 7, 10, 11, 12, 13].	Strong prior-art evidence.	This narrows the novelty claim. The article’s focus is the enterprise context-loss framing and the benchmarkable governance cases.
Dense-Mem can be used as one implementation of governed shared memory.	The package includes a deterministic graph baseline, vector and RAG answer-generation baselines, a Dense-Mem Docker scope-enforcement smoke test, a Dense-Mem fragment-recall answer-generation run, and a 100-scenario RAG-versus-knowledge-graph comparison.	Preliminary local evidence.	The 100-scenario comparison uses curated structured facts, not automatic fact extraction. It is not yet a full typed-fact Dense-Mem benchmark because the Dense-Mem recall run stores source fragments, not promoted enterprise facts with first-class supersession.

This boundary is important for the first Zenodo version. The article argues that governed shared memory is a necessary architectural layer to evaluate, not that the included local runs already prove production superiority over all RAG systems.

2. Problem Statement

This article focuses on a specific problem:

Enterprises need a governed memory layer because LLM-powered tools answer from incomplete context when organizational background knowledge is missing, stale, isolated, or inaccessible.

The problem is not that RAG is useless. RAG is useful. The problem is that many enterprise failures happen outside the narrow question ”did retrieval return a semantically similar chunk?”

2.1 Example: Business Context Missing From Developer AI

A product team decides that a billing feature is delayed until Q4 because a partner integration is not contractually approved. The decision is recorded in a planning chat and a meeting summary. A developer later asks a coding agent which billing work should be prioritized. If the coding agent only sees repository files and tickets, it may suggest implementing work that the business already deprioritized.

A memory layer should let the developer’s AI client read the approved business decision without giving it permission to mutate business records.

2.2 Example: Engineering Constraints Missing From Business AI

An engineering team records that a customer-facing endpoint cannot be exposed until authorization rules are redesigned. A business user later asks an AI assistant whether the company can promise that endpoint to customers. If the assistant only sees sales notes and public documentation, it may draft a commitment that engineering cannot safely deliver.

A memory layer should let a business AI client read the current technical constraint, cite the supporting engineering source, and indicate that older commitments are superseded.

2.3 Example: Automation Without Current Context

An automated agent closes support tickets when it finds a relevant product answer in an internal knowledge base. If the agent retrieves an older article whose answer has been replaced by a security policy, it may take an action that is semantically relevant but operationally wrong.

A memory layer should distinguish similarity from current truth.

3. What RAG Solves

RAG retrieves external information and makes it available to the model at generation time. This addresses a real limitation: the model’s pretrained parameters do not contain all private, current, or organization-specific knowledge. Lewis et al. framed RAG as combining parametric memory with a non-parametric memory accessed through retrieval [2].

In a typical enterprise RAG system:

1. Documents are collected from a knowledge base.
2. Documents are split into chunks.
3. Chunks are embedded into vectors.
4. A query is embedded into the same vector space.
5. Similar chunks are retrieved.
6. Retrieved chunks are added to the prompt.
7. The LLM generates an answer using the retrieved context.

This pattern is useful for many knowledge-intensive tasks. Enterprise RAG benchmarks such as EK-RAG and WixQA evaluate RAG systems against corporate or support-style question answering tasks [8, 9]. EnterpriseRAG-Bench moves closer to internal-company settings by generating a large synthetic corpus across enterprise source types and testing retrieval and reasoning capabilities, including conflict resolution and absence recognition [14].

4. What Vector Retrieval Does Not Inherently Solve

Vector retrieval ranks text by embedding similarity. This is powerful, but similarity is not the same as truth, authority, recency, permission, or organizational state.

The following table summarizes the gap.

Requirement	Vector RAG helps?	Remaining issue
Find semantically related text	Yes	Related text may not answer the actual operational question.
Use private documents	Yes	Access control must still be enforced outside the vector score.
Prefer current facts	Partly	Recency and supersession need explicit metadata or logic.
Detect conflicts	Partly	Similar chunks can conflict without the retriever knowing which is active.
Explain source authority	Partly	Retrieved chunks need provenance, ownership, and authority metadata.
Share memory across AI tools	Partly	Shared access requires identity, scopes, and isolation.
Prevent unauthorized mutation	No	Retrieval is not a write-governance model.

Longer context windows do not remove the need for these controls. Work on long-context evaluation shows that models can struggle with where relevant information appears in long inputs [3] and that effective context size can be smaller than advertised context length under more demanding tasks [5]. Chroma’s 2025 technical report likewise argues that input length can produce nonuniform model performance, especially when ambiguity and distractors are present [15]. These findings support a narrow claim: larger context can help, but larger context alone should not be treated as governed enterprise memory.

5. From Retrieval To Governed Memory

A graph-backed memory layer changes the question from ”which chunk is similar?” to ”what is the current, permitted, source-backed organizational fact?”

The memory layer should represent at least the following entities:

```
Team -> Profile/API key -> SourceFragment -> Claim -> Fact
      |                   |
      v                   v
    Source       Supersession
```

In this model:

- A `SourceFragment` preserves original evidence and provenance.

- A **Claim** represents a typed candidate assertion derived from evidence.
- A **Fact** represents a promoted active memory.
- A superseded fact remains available for audit instead of being overwritten.
- A team boundary controls which knowledge base is visible.
- A profile or API key controls which actor or AI client is reading or writing.
- A readonly key can recall knowledge without mutating it.
- A write key can submit evidence and candidate claims, subject to governance.

This does not replace RAG. It narrows RAG’s job. Vector search remains useful for candidate discovery, but the graph handles provenance, relationships, authority, status, and access policy.

Relationship types are a core benefit of the graph representation. A memory service can model that one source supports a fact, one fact supersedes another, two claims conflict, or a decision depends on a security or contract condition. When these relationships are maintained, retrieval can prioritize active, supported, higher-authority facts while preserving older superseded facts for audit. This is different from ranking chunks by similarity alone: a stale document can still be semantically close to the query, but an explicit supersession edge can mark it as no longer governing the answer.

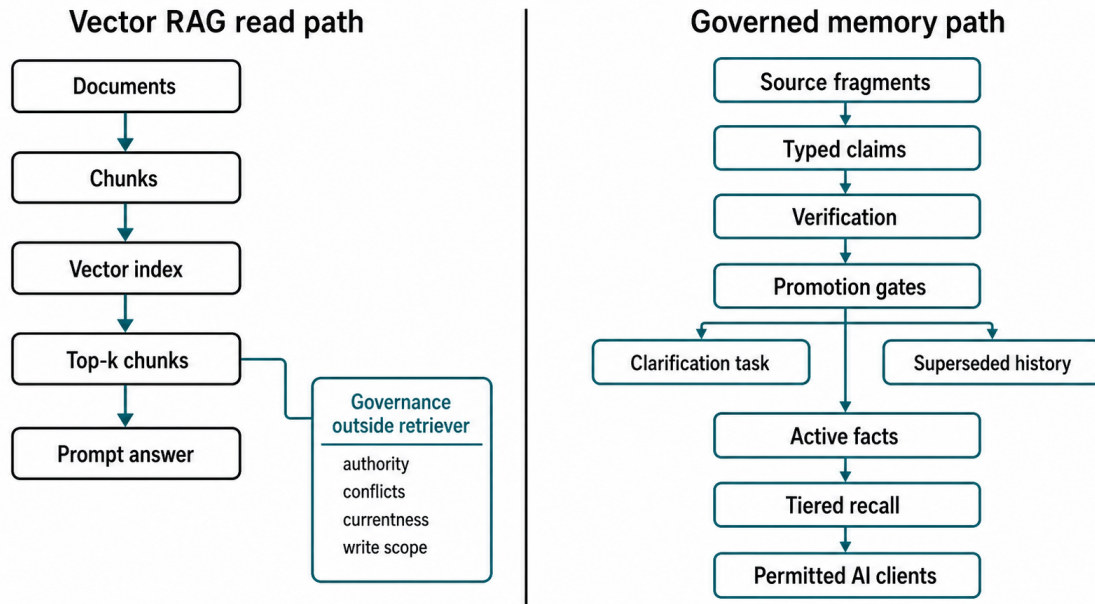


Figure 1: Vector RAG optimizes the read path from documents to retrieved chunks, while governed memory adds evidence storage, claim verification, promotion, permissions, conflicts, and supersession as explicit process stages.

6. Dense-Mem As A Reference Implementation

Dense-Mem is used here as a reference implementation for the architecture. It stores source fragments, typed claims, promoted facts, evidence links, verification metadata, and supersession history. It exposes MCP and REST surfaces so different clients can connect to a shared memory service. Its design separates the host LLM from memory governance: the host LLM remains responsible for conversation and extraction, while the memory service owns persistence, gates, audit metadata, recall, and access control [19].

The important implementation properties for this article are:

- Evidence is stored before promoted facts.
- Facts are not silently overwritten; corrections supersede older comparable facts.
- Comparable conflicts can return clarification tasks rather than letting the system choose silently.
- Knowledge is team-scoped.
- API keys can be read-scoped or write-scoped.
- The same memory service can serve multiple AI clients.
- Readonly clients can inspect permitted knowledge without mutating it.
- Operator and user portals are intentionally narrower than raw database access.

This enables cross-tool context sharing without collapsing all teams into one unrestricted memory space. A business user can connect a chat assistant with a readonly key to technical context. A developer can connect a coding agent with read access to approved business decisions. A separate write key can add business decisions, engineering constraints, or support incident facts into the same governed memory service.

6.1 Hosted Demo Scope

A hosted Dense-Mem demo is available at <https://demo-dense-mem.markhuang.ai> (<https://demo-dense-mem.markhuang.ai>). The demo creates a temporary isolated team and lets users test memory writes, recall, claims, verification, and fact promotion against a live reference instance before self-hosting. It should be treated as a disposable test environment, not as a place for secrets, personal data, credentials, production notes, or durable organizational records.

This matters for the article because the implementation is no longer only a local code artifact. Readers can inspect the behavior boundary directly: temporary team isolation, scoped API keys, MCP client configuration, and the difference between reading memory and mutating memory.

6.2 Process Difference From RAG

Dense-Mem differs from ordinary RAG most clearly at write time. RAG normally optimizes the read path: collect documents, split them into chunks, embed them, and retrieve similar chunks for a prompt. Governed memory adds a write-side lifecycle so the system can later explain why a fact is active, disputed, superseded, or unavailable.

Process question	Vector RAG process	Governed memory process
What is stored first?	Text chunks optimized for later similarity search.	Source fragments stored as evidence with provenance, source quality, labels, and scope.
What is extracted?	Extraction is optional and often left to the prompt or application layer.	The host extracts typed candidate claims with subject, predicate, object, confidence, modality, time bounds, and support links.
What is verified?	A retrieved chunk may be relevant, but relevance is not a claim-verification state.	A verifier evaluates whether a claim is supported by its evidence before it can become a durable fact.
What becomes current truth?	Currentness is usually encoded through metadata, recency filters, prompt rules, or application logic.	Promotion gates decide whether a validated claim can become an active fact.
What happens to conflicts?	Conflicting chunks can be retrieved together, but the retriever does not decide the active organizational fact.	Comparable conflicts become disputed claims and clarification tasks rather than silent overwrites.
What is recalled?	Usually top-ranked chunks.	Active facts, validated claims, and source fragments are returned as distinct authority tiers.
What controls writes?	Write control belongs to the surrounding application or database.	API scopes, team/profile isolation, promotion gates, and audit metadata are part of the memory service contract.

This distinction is why governed memory should not be evaluated only by retrieval accuracy. A system that retrieves a relevant stale document but cannot mark it superseded has solved a different problem from a system that can return the active fact and preserve the old document as evidence.

6.3 Promotion Logic

The high-level memory path stores evidence before it stores facts. A normal memory write creates a source fragment, creates one or more typed claims supplied by the host, verifies each claim against its supporting evidence, and promotes validated claims when auto-promotion is enabled. Bulk historical imports default to review first, so they can record evidence and claims without automatically turning every imported assertion into an active fact.

Promotion is deliberately stricter than "store the latest text." A claim must be validated, its predicate must be governed by an explicit promotion policy, and it must pass the predicate's gates. The gates combine extraction confidence, resolution confidence, modality, entailment, and support. Support is not a pure document-count rule: a claim can pass when it has enough supporting fragments or when a high-quality source satisfies the source-quality threshold.

Promotion policy	Process behavior	Example governance meaning
Single-current	Keep one active fact per subject and predicate. A stronger conflicting claim supersedes older facts; a comparable one becomes disputed; a weaker one is rejected.	A person's employer, a current policy owner, or a current project decision.
Multi-valued	Allow multiple active facts for the same subject and predicate.	Skills, tools used, known contacts, or preferences where several values can be true.
Versioned	Create a new fact version and close the older active version without treating the difference as an error.	Time-evolving organizational facts where history matters.
Append-only	Add immutable history without replacing prior facts.	Corrections, audit notes, or event-like memory.

For single-current facts, the contradiction path is the core governance difference. If a new validated claim has the same object as an existing active fact, it confirms the existing fact. If it conflicts and is stronger, it supersedes the older active fact while preserving the history. If it conflicts and is comparable, Dense-Mem marks the claim as disputed and returns a clarification task for the host to ask the user. If it is weaker, it is rejected instead of becoming active memory.

Promotion is also serialized per claim. This avoids duplicate fact creation when two clients try to promote the same claim at the same time. The design therefore treats promotion as a state

transition, not as a loose insertion into a vector index.

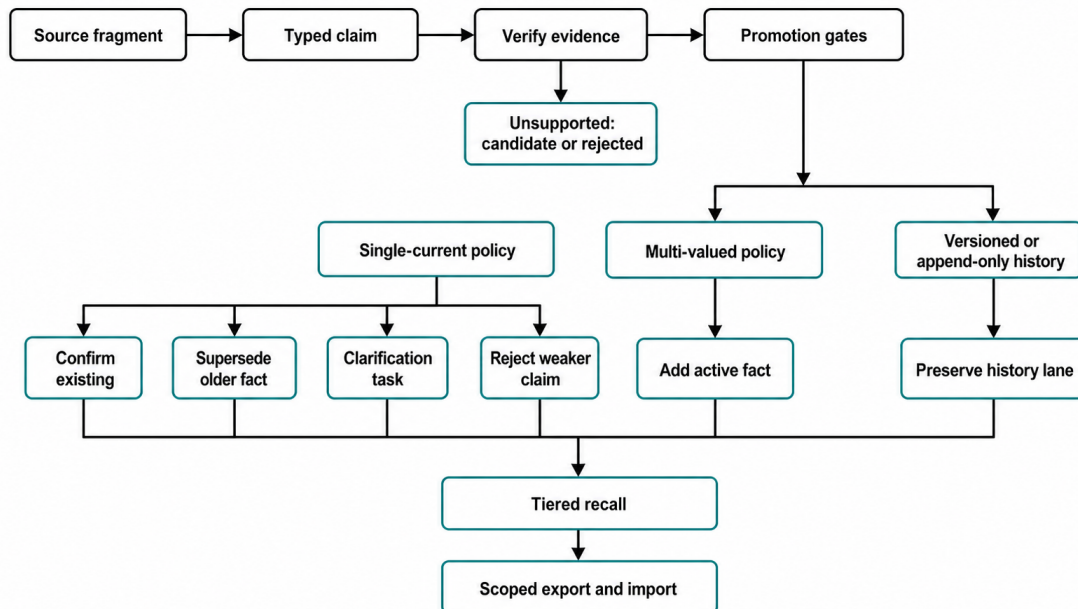


Figure 2: Dense-Mem promotion is a governed state transition: source fragments support typed claims; verified claims pass predicate gates; single-current conflicts are confirmed, superseded, disputed for clarification, or rejected; recall and export operate from the resulting governed states.

6.4 Recall, Export/Import, And Memory Portability

Recall is tiered. Active facts are the highest-authority results, validated claims are intermediate results, and source fragments remain available as raw evidence. Fragment recall still uses semantic and keyword retrieval, but those results are merged after profile filtering. This keeps vector search useful without making vector similarity the only authority signal.

Memory portability matters because useful agent behavior is not carried only by a skill file or tool instruction. A skill file can describe how to run a workflow, but memory carries the situated experience that makes the workflow perform well: what has been tried, which assumptions held, which background conditions mattered, what use case motivated a decision, which corrections were made, and what evidence supports the current answer. A team may therefore need to teach a newly connected agent not only a procedure, but also the governed knowledge and historical context that explain when that procedure should be used.

Export/import is the portability mechanism for that memory layer. It should move approved knowledge between AI clients without reducing it to opaque prompt text or an unreviewed database dump. A portable memory package can carry selected facts, validated claims, manual knowledge, and supporting evidence so the receiving system can inspect what it is being taught. This is closer to sharing an agent’s governed experience than sharing a static skill file.

Dense-Mem is one example of this pattern, not the boundary of the concept. Its available export path packages selected facts, validated claims, and manual triples into canonical JSON with a

SHA-256 integrity hash; when support is included, the package can carry supporting claims and source fragments so an imported item is not detached from its evidence. That example path is useful for controlled memory transfer, but the broader architectural target is portable governed memory across decisions, policies, temporal facts, background situations, use-case context, and accumulated operating experience.

Imports are designed as a governance workflow. Review-mode import records the bundle as evidence and creates claims without trusting them as active facts. Trusted import can validate and promote imported fact items, but trusted URL imports require an expected hash and conflicts require explicit decisions before local active facts are superseded. Import batches are recorded in a ledger so rollback can be attempted later; rollback is blocked if affected graph entities changed after the import. This favors auditable transfer over silent merge.

6.5 Performance And Operational Trade-Offs

The approach deliberately moves some cost from read-time prompting into write-time governance. Fragment creation may require embeddings, claim verification may call a verifier model, and promotion may traverse the graph, evaluate gates, serialize the claim transition, and write fact relationships. That is heavier than appending a note to a file or inserting a chunk into a vector database.

The benefit is that later reads have more structure to use. A query can return an active fact instead of asking the model to infer current truth from several chunks. A stale fact can remain visible as superseded history. A comparable conflict can become a user-facing clarification task. A readonly client can recall permitted knowledge while the storage layer rejects mutation.

The local benchmark results in Section 7.6 should be read through this trade-off. They suggest that explicit graph structure helps controlled conflict and multi-hop governance cases, but they do not yet prove production performance superiority for Dense-Mem. The next evidence step is a full typed-fact Dense-Mem ingestion benchmark that exercises promotion, supersession, conflict handling, and tiered recall as first-class Dense-Mem behavior.

7. Benchmark Design

This preprint defines a benchmark plan and includes executable seed-data checks. The current `data/results.jsonl` records local runs: a deterministic graph sanity baseline, a vector retrieval-only baseline, a metadata-filtered RAG answer-generation baseline, a Dense-Mem Docker smoke test, a Dense-Mem fragment-recall answer-generation run, and a 100-scenario RAG-versus-curated knowledge graph comparison. It does not yet report a full typed-fact Dense-Mem corpus-ingestion benchmark. The seed benchmark files in `data/` are a controlled starting point for the first experiment.

7.1 Compared Approaches

The first benchmark should compare four approaches:

1. Long-context prompting over a selected set of source documents.

2. Vector RAG using `text-embedding-3-large` with 3072-dimensional embeddings.
3. Graph and knowledge graph retrieval using explicit entity and relationship traversal.
4. Governed memory using source fragments, claims, active facts, supersession, team isolation, and read/write scopes.

The embedding model is fixed so the first study evaluates retrieval and memory governance behavior rather than embedding-provider variation.

7.2 Public Data Points

The benchmark package separates public evidence about enterprise AI adoption and existing RAG benchmarks from the synthetic enterprise task corpus. The file `data/public_data_points.jsonl` records curated data points such as Stanford AI Index and McKinsey adoption statistics, EnterpriseRAG-Bench corpus scale, WixQA dataset sizes, and Chroma’s long-context evaluation scope. These data points are not benchmark results for this article. They support the motivation and prior work framing.

7.3 Scenarios

The synthetic enterprise corpus should test:

- Current business decision recall.
- Superseded decision rejection.
- Conflicting claim detection.
- Missing-evidence refusal.
- Cross-team leakage prevention.
- Readonly mutation denial.
- Multi-client continuity between business and developer tools.
- Source-backed answer generation.

7.4 Metrics

The first benchmark should report:

- Answer correctness.
- Faithfulness to cited source.
- Supersession accuracy.

- Conflict handling accuracy.
- Unauthorized access rate.
- Unauthorized mutation rate.
- Context tokens used.

Security and governance metrics are first-class in this benchmark. A system that answers correctly by leaking another team’s knowledge should fail the benchmark.

7.5 Reproducibility Harness

The article package includes a local harness under `benchmark/` for validating seed data, generating the 100-scenario comparison corpus, collecting public-source retrieval evidence, scanning prior-art metadata, running a deterministic graph baseline, and running optional vector, RAG-answer, or Dense-Mem integration baselines when credentials are available. Optional runs must record unavailable dependencies as `not_run` rather than silently treating missing API keys or Docker services as successful benchmark results.

7.6 Preliminary Local Results

Six local runs were executed and recorded in `data/results.jsonl`. Dense-Mem runs were executed after resetting the benchmark Compose project and volumes.

Approach	Result	Interpretation
Deterministic governed graph sanity baseline	9/9 questions passed	The seed fact graph can represent current facts, supersession, disputes, missing evidence, access refusal, and readonly mutation refusal.
Permission-filtered vector retrieval baseline	7/9 questions passed	Vector retrieval found many supporting documents, but failed the conflict-detection case and the readonly-mutation case.
Metadata-filtered RAG answer-generation baseline	7/9 questions passed	RAG generated correct answers for most permitted-context tasks, but failed the conflict case after missing the authoritative product update and cannot enforce readonly mutation at the storage layer.
Dense-Mem Docker integration smoke test	Passed	A local Dense-Mem instance exposed read/write tools to a write-scoped key, hid write tools from a readonly key, and returned 403 for a readonly write attempt.
Dense-Mem fragment-recall answer-generation baseline	8/9 questions passed	Dense-Mem fragment recall plus answer generation handled the readonly mutation case through API scope enforcement, but still failed the conflict case because this run did not promote typed enterprise facts or model supersession as first-class Dense-Mem facts.

RAG versus curated knowledge graph comparison	RAG: 50/100; knowledge graph: 99/100	On 100 synthetic scenarios derived from the same scenario information, RAG over source documents passed simple lookup and supersession cases but failed the conflict-detection and multi-hop evidence cases under the combined retrieval-and-answer rubric. The curated knowledge graph baseline performed better because status, authority, conflict, supersession, and dependency edges were explicit.
-----------------------------------------------	-----------------------------------------	----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------

These results should be read narrowly. The graph baseline is deterministic and uses curated seed facts. The vector baseline scores retrieval only. The RAG answer-generation and Dense-Mem recall runs use an automated scoring rubric over synthetic documents and should be manually reviewed before being treated as publication-strength empirical results. The 100-scenario comparison recorded aggregate generation latency and token counts for audit: RAG used 154,887 ms, 23,654 prompt tokens, and 5,245 completion tokens, while the curated knowledge graph baseline used 125,545 ms, 27,651 prompt tokens, and 6,118 completion tokens. These local timings should not be generalized as performance claims without repeated runs and a fixed runtime environment. The 100-scenario comparison is measured benchmark data, but it is still synthetic and curated: the knowledge graph facts are generated from scenario templates rather than extracted automatically from documents. The single knowledge graph miss was an answer-classification error: the required facts were retrieved, but the model returned `answer_active` instead of `allowed_if_all_conditions` for one multi-hop case. The Dense-Mem recall run tests fragment recall and storage-scope enforcement, not a full typed-fact ingestion, promotion, supersession, or conflict-resolution benchmark.

8. Relationship To Existing Work

This article does not claim that graph retrieval is new. GraphRAG and knowledge graph-guided RAG are active areas. Microsoft’s GraphRAG approach constructs entity graphs and community summaries for query-focused summarization over private corpora [4]. Knowledge graph-guided RAG has been evaluated for improving retrieval and answer organization [7], and the GraphRAG survey literature frames graph-augmented retrieval as a broad design space [6].

This article’s focus is narrower and more operational:

- Enterprise memory must connect multiple AI clients.
- Shared memory must preserve evidence and fact history.
- Read and write permissions are not optional deployment details.
- Stale facts and conflicting facts must be represented explicitly.
- Governance failures should be benchmarked alongside answer quality.

Zep’s temporal knowledge graph architecture for agent memory is especially relevant prior work because it emphasizes dynamic enterprise knowledge, conversation-derived memory, and temporal

graph structure [10]. More recent enterprise-agent memory work is even closer. Taheri’s Governed Memory preprint frames a shared memory and governance layer for multi-agent workflows and reports controlled experiments on governance routing, token reduction, cross-entity leakage, and governed-memory saturation [11]. Chamarty’s Context Objects proposal treats temporal validity, provenance lineage, confidence, organizational weight, and feedback as first-class retrieval properties for enterprise AI agents [12]. Srinivasan’s Stateless Decision Memory argues that regulated enterprise agents require deterministic replay, auditable rationale, multi-tenant isolation, and statelessness for horizontal scale [13].

These works narrow the novelty claim of this article. The contribution here is not “governed memory exists” or “temporal enterprise memory is new”; it is a pedagogical and benchmark-oriented framing of the enterprise context-loss problem, with Dense-Mem used as one reference implementation for permissioned knowledge graph memory, fact supersession, conflicts, and cross-client access control.

9. Discussion

RAG moves LLM systems beyond pretrained knowledge, but RAG by itself is not a complete enterprise memory model. A vector database can retrieve similar text, but enterprises also need to know who can access the text, whether the fact is current, where the fact came from, whether a newer fact supersedes it, and whether two sources conflict.

The opportunity is to make shared memory an enterprise infrastructure layer. Instead of every AI tool starting from a cold prompt or a disconnected knowledge-base index, tools can connect to a governed memory service. This does not require every tool to use the same model or the same user interface. It requires a shared memory contract:

```
read permitted facts
cite source evidence
respect team boundaries
reject unauthorized mutation
surface conflicts
preserve supersession history
```

This framing builds on the author’s prior informal discussion of AI memory beyond RAG [18], but this article narrows the claim toward an enterprise-governance architecture and a benchmarkable evaluation plan.

For enterprise automation, this matters because actions depend on current context. The goal is not to make AI systems omniscient. The goal is to make the available organizational context explicit, source-backed, permissioned, and auditable.

10. Limitations

This preprint currently defines the architecture, benchmark plan, and local baselines. It includes preliminary answer-generation runs and a 100-scenario RAG-versus-curated-knowledge-graph comparison, but it does not yet include a full typed-fact Dense-Mem ingestion benchmark. The initial

corpus is synthetic so that stale facts, conflicts, team boundaries, and access-control cases can be controlled. Synthetic data weakens claims about real-world frequency, but it strengthens evaluation control for governance failure modes.

The article also does not yet quantify how often context is lost across specific enterprise AI products, such as office copilots, coding agents, chat assistants, and workflow agents. It does not compare the cost of governed shared memory with custom model training. Those questions matter, but they require a different survey or deployment study than the seed benchmark included here.

Future versions should add:

- A full typed-fact Dense-Mem ingestion benchmark that exercises fact promotion, supersession, conflict resolution, and relationship-aware recall as first-class Dense-Mem behavior.
- A stronger default path from bulk source fragments to typed enterprise claims, because the current Dense-Mem recall baseline still evaluates fragment recall rather than complete fact promotion over the benchmark corpus.
- Broader memory export/import coverage for enterprise decisions, policy facts, temporal facts, background situations, use-case context, and accumulated operating experience.
- Additional model families and repeated benchmark runs if generation behavior becomes a study variable.
- A real-world case study where data can be shared without exposing private organizational information.

11. Conclusion

Enterprise AI systems need more than larger context windows and isolated RAG indexes. They need governed shared memory: source-backed, permissioned, team-isolated, and capable of representing fact history. Vector retrieval is a useful component, but similarity alone is not current organizational truth. A knowledge graph memory layer can make facts, sources, teams, actors, conflicts, and supersession explicit. That is the core opportunity for scalable enterprise AI systems: connect different AI clients through governed memory so automation and human-facing assistants can reason from the same permitted organizational context.

Revision History

- v0.1, 2026-05-30: Initial preprint with the governed-memory architecture, benchmark plan, seed datasets, and local baseline results.
- v0.2, 2026-06-02: Added hosted demo context, expanded the process discussion from source fragments through claims, verification, promotion, recall, and export/import, reframed memory portability as governed knowledge and experience transfer, added process-flow figures, and clarified benchmark and portability limitations.

References

- [1] Vaswani, Ashish; Shazeer, Noam; Parmar, Niki; Uszkoreit, Jakob; Jones, Llion; Gomez, Aidan N.; Kaiser, Lukasz; Polosukhin, Illia (2017). *Attention Is All You Need*. Advances in Neural Information Processing Systems. <https://arxiv.org/abs/1706.03762>.
- [2] Lewis, Patrick; Perez, Ethan; Piktus, Aleksandra; Petroni, Fabio; Karpukhin, Vladimir; Goyal, Naman; Kuttler, Heinrich; Lewis, Mike; Yih, Wen-tau; Rocktaschel, Tim; Riedel, Sebastian; Kiela, Douwe (2020). *Retrieval-Augmented Generation for Knowledge-Intensive NLP Tasks*. Advances in Neural Information Processing Systems. <https://arxiv.org/abs/2005.11401>.
- [3] Liu, Nelson F.; Lin, Kevin; Hewitt, John; Paranjape, Ashwin; Bevilacqua, Michele; Petroni, Fabio; Liang, Percy (2024). *Lost in the Middle: How Language Models Use Long Contexts*. Transactions of the Association for Computational Linguistics. <https://arxiv.org/abs/2307.03172>.
- [4] Edge, Darren; Trinh, Ha; Cheng, Newman; Bradley, Joshua; Chao, Alex; Mody, Apurva; Truitt, Steven; Metropolitansky, Dasha; Ness, Robert Osazuwa; Larson, Jonathan (2024). *From Local to Global: A Graph RAG Approach to Query-Focused Summarization*. arXiv:2404.16130. <https://arxiv.org/abs/2404.16130>.
- [5] Hsieh, Cheng-Ping; Sun, Simeng; Krizan, Samuel; Acharya, Shantanu; Rekesh, Dima; Jia, Fei; Zhang, Yang; Ginsburg, Boris (2024). *RULER: What's the Real Context Size of Your Long-Context Language Models?*. arXiv:2404.06654. <https://arxiv.org/abs/2404.06654>.
- [6] Han, Haoyu; Wang, Yu; Shomer, Harry; Guo, Kai; Ding, Jiayuan; Lei, Yongjia; Halappanavar, Mahantesh; Rossi, Ryan A.; Mukherjee, Subhabrata; Tang, Xianfeng; He, Qi; Hua, Zhigang; Long, Bo; Zhao, Tong; Shah, Neil; Javari, Amin; Xia, Yinglong; Tang, Jiliang (2025). *Retrieval-Augmented Generation with Graphs (GraphRAG)*. arXiv:2501.00309. <https://arxiv.org/abs/2501.00309>.
- [7] Zhu, Xiangrong; Xie, Yuexiang; Liu, Yi; Li, Yaliang; Hu, Wei (2025). *Knowledge Graph-Guided Retrieval Augmented Generation*. Proceedings of the 2025 Conference of the Nations of the Americas Chapter of the Association for Computational Linguistics: Human Language Technologies. DOI: <https://doi.org/10.18653/v1/2025.naacl-long.449>. <https://aclanthology.org/2025.naacl-long.449/>.
- [8] Yu, Tan; Zhou, Wenfei; Yang, Lei; Shukla, Aaditya; Madugula, Meenakshi; Gundecha, Pritam; Burnett, Nick; Xu, Anbang; Seth, Vishal; Bar, Tamar; Akkiraju, Rama; Zhang, Vivienne (2025). *EKRAG: Benchmark RAG for Enterprise Knowledge Question Answering*. Proceedings of the 4th International Workshop on Knowledge-Augmented Methods for Natural Language Processing. DOI: <https://doi.org/10.18653/v1/2025.knowledgenlp-1.13>. <https://aclanthology.org/2025.knowledgenlp-1.13/>.
- [9] Cohen, Dvir; Burg, Lin; Pykhnivskiy, Sviatoslav; Gur, Hagit; Kovynov, Stanislav; Atzmon, Olga; Barkan, Gilad (2025). *WixQA: A Multi-Dataset Benchmark for Enterprise Retrieval-Augmented Generation*. arXiv:2505.08643. <https://arxiv.org/abs/2505.08643>.
- [10] Rasmussen, Preston; Paliychuk, Pavlo; Beauvais, Travis; Ryan, Jack; Chalef, Daniel (2025). *Zep: A Temporal Knowledge Graph Architecture for Agent Memory*. arXiv:2501.13956. <https://arxiv.org/abs/2501.13956>.

- [11] Taheri, Hamed (2026). *Governed Memory: A Production Architecture for Multi-Agent Workflows*. arXiv:2603.17787. <https://arxiv.org/abs/2603.17787>.
- [12] Chamarty, Madhu (2026). *Context Objects: A Temporal, Provenance-Aware Memory Primitive for Enterprise AI Agents*. DOI: <https://doi.org/10.2139/ssrn.6775102>. https://papers.ssrn.com/sol3/papers.cfm?abstract_id=6775102.
- [13] Srinivasan, Vasundra (2026). *Stateless Decision Memory for Enterprise AI Agents*. arXiv:2604.20158. <https://arxiv.org/abs/2604.20158>.
- [14] Sun, Yuhong; Rahmfeld, Joachim; Weaver, Chris; Chen, Weijia; Desai, Roshan; Huang, Wenxi; Butler, Mark H. (2026). *EnterpriseRAG-Bench: A RAG Benchmark for Company Internal Knowledge*. arXiv:2605.05253. <https://arxiv.org/abs/2605.05253>.
- [15] Hong, Kelly; Troynikov, Anton; Huber, Jeff (2025). *Context Rot: How Increasing Input Tokens Impacts LLM Performance*. Chroma. <https://www.trychroma.com/research/context-rot>.
- [16] McKinsey & Company (2025). *The State of AI in 2025: Agents, Innovation, and Transformation*. McKinsey & Company. <https://www.mckinsey.com/capabilities/quantumblack/our-insights/the-state-of-ai>. See key findings item 3, section "Many organizations are already experimenting with AI agents," and section "AI as a catalyst for innovation."
- [17] Stanford Institute for Human-Centered Artificial Intelligence (2026). *The 2026 AI Index Report*. Stanford University. <https://hai.stanford.edu/ai-index/2026-ai-index-report>. See Economy chapter, item 5.
- [18] Huang, Zhirong (2026). *AI Memory Beyond RAG*. <https://markhuang.ai/blog/ai-memory-beyond-rag>. Informal blog post.
- [19] Huang, Zhirong (2026). *Dense-Mem*. <https://github.com/markhuangai/dense-mem>. Software repository and gray literature.