FlowX.AI

TECHNICAL PAPER

MNEME · ACTIVE, GOVERNED MEMORY FOR AI AGENTS

A Knowledge Graph Agents Can Write To, Safely

MNEME, the FlowX.AI Ontology Layer: a provenance-tracked structured-memory graph that agents read, validate against, and contribute to, without corrupting the institutional source of truth.

Bogdan Răduță
Head of Research, FlowX.AI
bogdan.raduta@flowx.ai

FlowX.AI · Applied Research

August 2026 · Version 1.1

ABSTRACT

The 2026 consensus in enterprise AI is that retrieval is not enough: agents need structured, relational, persistent knowledge, and the knowledge graph has become the "beyond-RAG" layer of choice, with graph retrieval reported at 86% multi-hop accuracy against 32% for vector search alone[2]. Yet across the field one assumption is nearly universal: the ontology is *read-only* context the agent consults, and keeping it correct is a separate human discipline. We present **MNEME**, the FlowX.AI Ontology Layer, which makes the opposite choice: the ontology is *active, governed memory the agent helps build*. Agents read, validate against, and write back to the same project-scoped knowledge graph, and every contribution is safe by construction. Each Concept and Relation carries *provenance* (pdm_import, agent_write, or manual) and *confidence* as first-class fields; agent writes are staged below a confidence bar, screened by conflict detection of three kinds, and quarantined out of canonical retrieval until promoted by reinforcement across runs or by human review. The institutional source of truth, imported one-way from the platform data model, is authoritative and read-only to agents, so a hallucination can never become a regulatory taxonomy entry, while genuinely new structure is captured rather than lost. MNEME runs on two pluggable backends behind one contract, Postgres with pgvector for semantic-heavy ontologies and Neo4j for graph-heavy multi-hop, retrieves by a hybrid of vector similarity and graph expansion in a single call, and exposes a nine-node toolkit on the Agent Builder canvas. We describe the architecture, the safe write-back loop, the toolkit and its agent patterns, an illustrative regulated taxonomy, and how active memory differs from the read-only knowledge graphs the rest of the market ships.

KEYWORDS ontology · knowledge graph · active memory · provenance · GraphRAG · entity resolution · conflict detection · structured memory

1 Introduction

An agent that answers from vector search alone is working from a pile of look-alike passages with no notion of how anything relates to anything else. The enterprise has noticed, and 2026 has been the year the knowledge graph moved from a data-team artifact to the centerpiece of agent design: graph retrieval reaches multi-hop accuracy that vector search cannot approach[2], analysts name semantic layers and GraphRAG among the year's defining trends, and platform vendors now ship ontology-driven agent construction as a headline feature. The premise is sound. Agents need persistent, relational, traceable knowledge, and an ontology supplies exactly the structure that flat retrieval lacks[1, 5].

But look at how these systems treat the ontology and a shared limitation appears: it is a *read-only* oracle. The agent queries it, grounds an answer in it, and moves on; the graph itself is maintained by a separate human governance process, on a separate cadence, with the agent strictly a consumer. That is a strange division of labor for a system whose entire promise is that it learns. The agent that just resolved an ambiguous entity, classified a novel document, or discovered a relationship the graph did not contain has produced exactly the knowledge the ontology most needs, and then throws it away. The institutional memory does not improve from use; it improves only when a human gets around to editing it.

> *Everyone makes the ontology a read-only oracle the agent consults. The agent that just learned something the graph lacks is the agent best placed to extend it, and the one whose contribution we are most afraid to trust. Make the ontology active memory, and make the write safe.*
>
> MNEME · CORE THESIS

The reason the field keeps the ontology read-only is not oversight; it is fear, and a reasonable one. In a regulated enterprise the ontology encodes product taxonomies, regulatory classifications, and risk hierarchies that must stay consistent across thousands of agent runs. Letting a stochastic model write to that structure sounds like inviting a hallucination into the source of truth. We present **MNEME**, the FlowX.AI Ontology Layer, which resolves the dilemma rather than ducking it: agents *do* write back, but every contribution is provenance-stamped, conflict-screened, staged below a confidence bar, quarantined from canonical retrieval, and promoted only by repeated corroboration or human review. The authoritative facts, imported from the platform data model, remain read-only to agents. The result is active memory that gets richer every run, with the regulatory blast radius of a read-only system.

1.1 Contributions

- A reframing of the enterprise ontology from a read-only oracle to active, governed memory that agents read, validate against, and safely extend.
- A provenance-first domain model in which every Concept and Relation carries its trust pedigree and confidence as first-class fields, making the graph auditable by construction.

- A safe write-back loop: deduplicate, conflict-detect, stage below a confidence bar, quarantine from canonical retrieval, and promote by reinforcement or human review.
- A dual-backend architecture (Postgres with pgvector, or Neo4j) behind one contract-tested interface, with hybrid vector-plus-graph retrieval in a single call.
- A nine-node Agent Builder toolkit that makes ontology operations first-class on the visual canvas, and the agent patterns that compose them.

2 Read-Only Knowledge, and Its Limits

2.1 Three kinds of agent knowledge

Enterprise agents draw on three distinct stores, and conflating them is the source of much confusion. *Retrieval* (RAG over a vector store) supplies unstructured content by similarity, excellent for "what does this document say" and blind to structure. *Operational data* (databases, the platform data model) supplies the authoritative current state of entities, excellent for "what is this customer's balance" and not designed for semantic reasoning. Between them sits the *ontology*: a graph of concepts and relations that says how the business's world is organized, what a Mortgage is, that it is a Secured Loan, that a Secured Loan inherits certain required attributes. Graph retrieval over that structure is what enables multi-hop reasoning and explainable paths that flat similarity cannot produce[2,5].

2.2 The read-only assumption, and what it costs

The market has converged on building that middle layer, but on treating it as immutable from the agent's side. The cost is twofold. First, the ontology decays relative to reality: new products, new regulations, and newly-resolved entity variants accumulate in agent runs and are never written back, so the graph drifts out of date until a human governance cycle catches up. Second, the knowledge an agent is uniquely positioned to capture, that two records are the same entity, that a document introduces a concept the taxonomy lacks, is exactly the knowledge that evaporates at the end of the run. A system sold as institutional memory does not actually remember anything its agents learn.

2.3 Why write-back is hard, and what makes it safe

The reason no one lets agents write is that the obvious implementation is dangerous: if an agent can mint a concept, a hallucination becomes a taxonomy entry, and in a regulated setting that is a compliance incident waiting to propagate. Safe write-back is therefore not a feature toggle but an architecture: contributions must be attributable (provenance), screened against the existing structure (conflict detection), held apart from authoritative knowledge until they earn their place (staging and promotion), and unable to affect canonical answers while pending (quarantine). MNEME is the design of that architecture. Table 1 contrasts the read-only norm with active memory.

Table 1. Read-only knowledge graph versus active, governed memory.

Property	Read-only KG (the norm)	MNEME (active memory)
Agent role	Consumer only	Consumer and contributor
Improves from use?	No (human edits only)	Yes (staged, promoted)
Provenance	Often absent or metadata	First-class on every fact
Hallucination risk to canon	N/A (no writes)	Quarantined; cannot reach canon
Conflict handling	Manual	Detected and logged pre-stage
Source of truth	Separate, manual	PDM imported, read-only to agents

3 The MNEME Architecture

MNEME sits as a structured-memory layer between unstructured retrieval and operational data. It does not replace either: the knowledge base still holds document chunks, and the platform data model still holds authoritative entity state. The ontology is the connective structure agents reason over and contribute to. Figure 1 shows the three layers and the flows between them.

FIG. 1 · THE STRUCTURED-MEMORY LAYER BETWEEN RETRIEVAL AND OPERATIONAL DATA

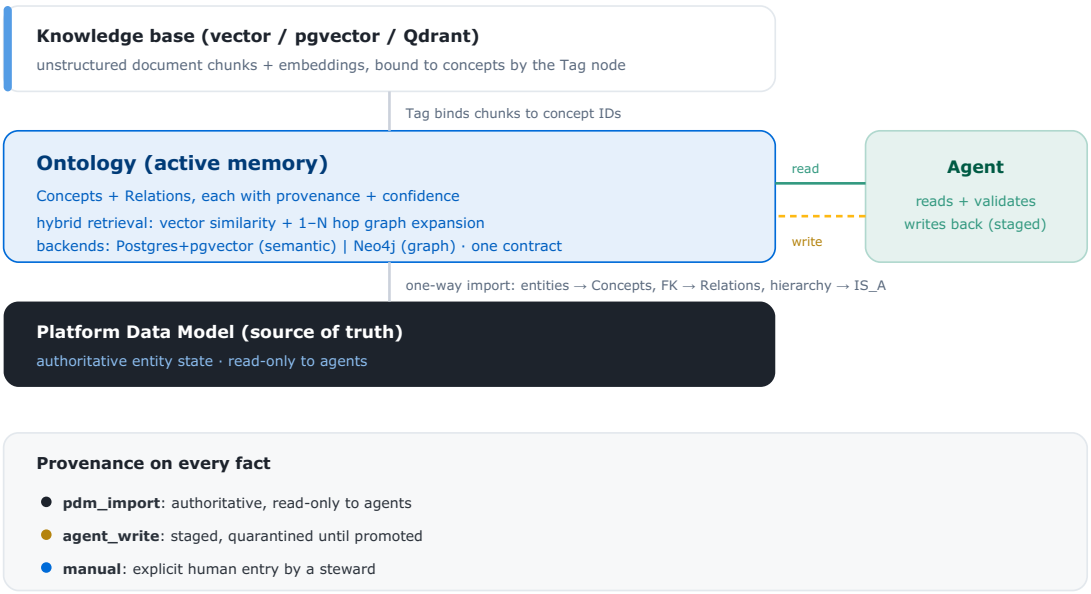

Figure 1. The ontology is the middle layer: the knowledge base supplies content (bound to concepts by the Tag node), the platform data model supplies authoritative state (imported one-way, read-only to agents), and the agent both reads from and writes back to the ontology. Every fact carries provenance, which is what makes the read and write semantics safe to differentiate.

3.1 A provenance-first domain model

The unit of knowledge is a Concept (with a label, synonyms, an IS_A position, and an attribute schema) or a Relation between concepts. What makes the model unusual is that *provenance* and *confidence* are not metadata bolted on the side but first-class fields every fact must carry. A fact is one of three provenances: *pdm_import* (drawn from the authoritative data model, and read-only to agents), *agent_write* (proposed by an agent, and quarantined until promoted), or *manual* (entered explicitly by a human steward). Read and write permissions follow from provenance, so the system can let agents enrich the graph while guaranteeing they cannot alter the authoritative core. An auditor can reconstruct, for any concept, who asserted it, when, and with what authority.

3.2 Two backends, one contract

Ontologies differ in shape. A product catalog is semantic-heavy and well served by vector similarity over Postgres with pgvector; a regulatory hierarchy is graph-heavy and rewards the native multi-hop traversal of Neo4j. MNEME supports both behind a single *OntologyStore* interface whose behavioral

parity is enforced by contract tests run against both real backends in CI, so a team chooses the backend that fits the workload rather than the one the platform happened to pick, and can switch without rewriting flows. Embeddings are computed locally with a compact sentence-transformer model, so the layer runs in air-gapped deployments with no outbound calls.

3.3 Hybrid retrieval

A query against the ontology is not a pure nearest-neighbor lookup. MNEME combines vector similarity with graph expansion of one to N hops in a single call, and ranks results by a blend of similarity and popularity, so an agent grounds its answer in context that is both semantically and relationally relevant. This is the operational form of the GraphRAG advantage[2]: the agent does not merely find the closest passage, it finds the connected neighborhood of concepts that bear on the question, and can cite the path.

4 Active Memory: the Safe Write-Back Loop

The defining capability, and the one the rest of the market does not offer, is that an agent can contribute to the graph without endangering it. Write-back is a pipeline, not a single operation, and every stage exists to make a contribution safe. Figure 2 shows it.

FIG. 2 · HOW AN AGENT CONTRIBUTION BECOMES CANONICAL KNOWLEDGE

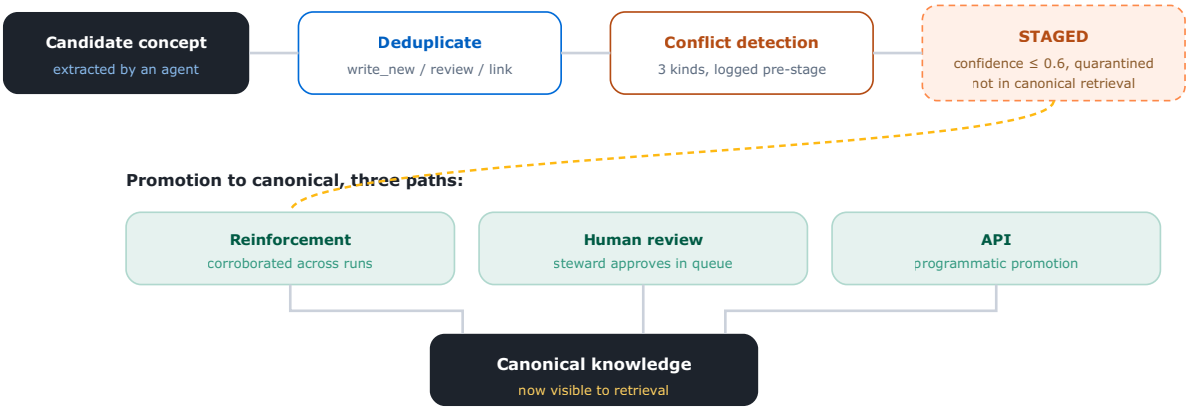

Figure 2. A candidate concept is first deduplicated (so an agent rediscovering an existing concept under a new phrasing links rather than duplicates), then screened for three kinds of conflict, then staged below a confidence bar where it is durable but invisible to canonical retrieval. It becomes canonical only by reinforcement across runs, explicit human review in the pending queue, or a programmatic promotion. A hallucination that is never corroborated and never approved simply expires in staging.

4.1 Deduplicate before writing

The most common failure of agent write-back is not a hallucination but a near-duplicate: an LLM rediscovers *Personal Loan* as *Retail Personal Loan* and proposes it as new. The Deduplicate step runs a pre-flight similarity check and returns a recommended action, write a new concept, route for review, or link to the existing one, so a write happens only when the candidate genuinely adds structure. This keeps the graph from bloating with synonyms, which is the quiet way an agent-writable ontology would otherwise rot.

4.2 Conflict detection and staging

A surviving candidate is screened for three kinds of conflict against the existing structure, a label clash, a category mismatch, and a relation that contradicts an existing one, each detected and written to a queryable log before anything is staged. The contribution is then staged at low confidence: durable, attributable, and visible in a pending-writes queue, but excluded from canonical retrieval, so it cannot influence a production answer while it waits. This is the same quarantine logic that hallucination containment applies to outputs, here applied to memory: the unverified thing is kept where it can do no harm until it is verified.

4.3 Promotion earns canonical status

A staged contribution becomes canonical only by earning it, along one of three paths. *Reinforcement* promotes a concept that is independently rediscovered across enough runs, on the principle that corroboration is evidence. *Human review* lets a domain steward approve or reject from the pending queue, the path a regulated taxonomy change will usually take. *Programmatic promotion* via the API supports controlled bulk workflows. A contribution that is never corroborated and never approved never becomes canonical; it expires harmlessly in staging. Validation with IS_A inheritance runs throughout, so a promoted concept must satisfy the required attributes of its parents, halting any agent output that violates the schema before it reaches a downstream system.

5 The Nine-Node Toolkit

On the Agent Builder canvas, MNEME appears as a dedicated Ontology palette of nine nodes, each mapping to one operation an agent performs against structured memory. They are drop-in: each writes well-known keys to the workflow state, ready for prompt injection or downstream branching, with no glue code.

Node	What it does	Common use
Retrieve	Hybrid semantic + graph retrieval; top-K concepts with similarity, popularity, optional 1–N hop expansion	Grounding a prompt with relevant ontology context
Classify	Single-label classification: text to top concept + confidence	Flow routing and branching on a matched concept
Describe	Lookup by ID with full grounding context (label, synonyms, IS_A chain, attribute schema)	Deterministic grounding when the concept is known
Expand	Walk the relation graph from a concept (for example all IS_A descendants to N hops)	"Find all loan products that inherit from Secured Loan"
Tag	Tag a longer text with concept IDs by chunking and per-chunk semantic resolution	Document ingestion: concept tags as KB metadata
Deduplicate	Pre-flight similarity check before a write; returns write_new / review / link_to_existing	Prevent memory-write spam from rephrased rediscoveries
Validate	Validate a structured payload against ontology rules with IS_A inheritance	Halt or warn on agent output that violates schema
Memory Write	Stage a candidate concept or relation; conflicts detected and logged	Agents propose new structure; humans review
Memory (all-in-one)	Multi-mode node combining retrieve / validate / memory-write in one config	Compact flows where one node runs the whole loop

5.1 The patterns that compose them

A handful of compositions cover most real flows. **Retrieval-grounded answer:** Retrieve, then answer with the returned context injected into the prompt. **Classify then describe:** Classify the input to a concept and, on a match, Describe it for clean grounding, or ask a clarifying question when nothing crosses the similarity threshold. **Multi-hop reasoning:** Classify to a starting concept, Expand its IS_A family in one call, and let the model synthesize over the family, the Neo4j differentiator made tangible. **Validate before commit:** have the model emit structured output, then Validate it against the schema so an invalid payload halts before any downstream effect. **Contribute with rails:** extract a candidate, Deduplicate it, and only Memory-Write the genuinely new, staged for review. The same nine nodes serve a read-only grounding flow and a full read-reason-validate-contribute loop; the difference is which nodes the author places.

6 The Document-to-Ontology Bridge

A recurring question is how unstructured documents map onto the ontology. The answer is two layers, two indexes, one binding, and it is deliberately not "put the documents in the graph." The knowledge base holds the document chunks and their embeddings; the ontology holds the concepts; and the Tag node is the binding, resolving a document's text against the ontology and attaching the resulting concept IDs as metadata on the stored chunks. A later query can then filter the knowledge base by ontology concept, "find chunks tagged under this regulatory category," rather than by raw similarity alone, so the document store becomes navigable by meaning. Where a document genuinely introduces a new concept, a gated Memory Write stages it for review. The ontology is never the document store; it is the structural index that makes the document store comprehensible, and the Tag node keeps the two in registration without an agent having to manage the link by hand.

By default the Tag node is pure-semantic, chunk then resolve then threshold, with no model call, which makes it fast, free, and deterministic. Where higher-quality entity extraction is needed, an LLM extraction step can precede it, emitting a list of mentions that Tag resolves one by one; the node accepts that list directly, so the upgrade is a one-node composition rather than a rebuild.

7 Illustrative Application

Consider the loan-product taxonomy of an illustrative regulated bank, Meridian (a fictional example used for continuity across this series). The ontology encodes the hierarchy, Mortgage is a Secured Loan is a Loan, with required attributes inherited along the IS_A chain, and is backed by Neo4j because the reasoning is graph-heavy.

In day-to-day grounding, an underwriting agent Classifies an incoming application to a product concept, Expands the concept's governing rules in a single multi-hop call, and Validates that the proposed terms inherit the required attributes of the parent type; a violation halts the flow before terms reach a downstream system. None of this writes to the graph, it is read-only grounding, and it already exceeds what flat retrieval could do, because the agent reasons over the product family, not over look-alike paragraphs.

The active-memory difference shows when a new regulation is published. An agent extracts the new classification it introduces, runs Deduplicate (which confirms it is genuinely new, not a rephrasing of an existing rule), and issues a Memory Write. Conflict detection checks the candidate against the existing hierarchy and finds no contradiction, so it is staged at low confidence, attributed to *agent_write*, and held in the pending-writes queue, invisible to production grounding. A compliance steward reviews it there, confirms the placement in the taxonomy, and promotes it; only now does it become canonical and start grounding answers, carrying its provenance for any future audit. The institutional taxonomy improved from an agent run, on a regulated artifact, without a single moment at which an unreviewed model output could have driven a lending decision. That is the whole point of making memory active and governed at the same time.

8 Empirical Evaluation

Section 3.3's hybrid-retrieval claim, and the 86% vs. 32% multi-hop figure it cites from GraphRAG[2], describes a general property of combining vector search with graph traversal, not a number MNEME itself has published. To check whether that property holds for the specific retrieval mechanism this paper describes (a vector-seeded entry point plus a one-hop expansion over entity-mention links, Section 3.3), we built a standalone reference implementation and evaluated it on a public multi-hop QA benchmark. The implementation is a minimal reproduction of the mechanism, not the FlowX.AI Platform's production Postgres/Neo4j retrieval path (which uses learned embeddings and a persisted, corpus-wide graph rather than a per-query mention graph); it exists to test the claim, and the code and test suite are released as supplementary material.

8.1 Dataset and setup

We use the HotpotQA distractor validation set[13]: 7,405 questions, each with ten candidate Wikipedia paragraphs (two containing the supporting facts, eight textually plausible distractors) and human-annotated supporting-fact titles. We compare two retrieval modes with an identical entry point: *vector-only* ranks the ten candidates by BM25 similarity to the question and takes the top-k; *hybrid* takes the top-1 BM25 candidate as an entry point, builds a per-question graph with an edge between any two candidates where one paragraph's text mentions the other's title, and returns the entry point plus its graph neighbors (falling back to the next-best BM25 candidates if the budget k is not filled by neighbors alone). A retrieval is scored as a hit only if both gold supporting titles appear in the returned set (exact-recall@k), evaluated separately for HotpotQA's two reasoning types: *bridge* questions, which require traversing from one entity to a second, textually unrelated one, and *comparison* questions, which typically name both entities directly.

8.2 Results

Table 1 reports exact-recall@k over the full validation set. On bridge questions, the reasoning type the graph hop specifically targets, hybrid retrieval roughly doubles vector-only recall at both budgets (@2: 55.5% vs. 27.4%; @4: 74.5% vs. 51.1%), directionally consistent with the paper's cited claim that graph traversal recovers multi-hop evidence a similarity search alone misses. On comparison questions, where both entities are typically already named in the question and vector search alone can find both paragraphs directly, the graph hop does not help and is slightly negative (@2: 15.5% vs. 20.4%): the entry-point-plus-neighbor heuristic can spend its budget on a graph neighbor of the top BM25 hit instead of the second named entity, when the two compared entities do not mention each other. Both results are the honest shape of the mechanism: it wins specifically where the paper says the market's read-only, vector-only graphs fall short, and does not universally dominate.

Table 1. Exact-recall@k (both gold supporting titles retrieved), HotpotQA distractor validation, n=7,405.

Question type	n	Vector-only@2	Hybrid@2	Vector-only@4	Hybrid@4
Bridge (multi-hop)	5,918	27.4%	**55.5%**	51.1%	**74.5%**
Comparison	1,487	**20.4%**	15.5%	46.3%	46.1%
Overall	7,405	26.0%	**47.5%**	50.1%	**68.8%**

8.3 What this evaluation does and does not show

This experiment isolates the retrieval mechanism of Section 3.3 in a standalone harness built for this test; it does not exercise the write-back pipeline of Section 4, the provenance model, or the real Postgres/Neo4j backends, none of which HotpotQA is designed to evaluate. The result supports a narrower claim than "hybrid retrieval is better": it shows that vector-seeded graph expansion helps precisely on the multi-hop reasoning type it targets, and can slightly hurt on a reasoning type it does not, which is the kind of result an honest evaluation should produce rather than a uniform win. Benchmarking the write-back loop (promotion accuracy, false-promotion rate under the confidence-staging policy of Section 4) against a labeled corroboration dataset remains future work. The reference implementation, the HotpotQA evaluation script, and a pytest suite that reproduces Table 1 are released as supplementary material.

9 Discussion

The market is right that agents need a knowledge graph; the question MNEME answers differently is whether the agent is allowed to help build it. Read-only graphs, including the ontology-driven agent offerings shipping in 2026[7], treat maintenance as a human discipline orthogonal to agent operation, which leaves the graph perpetually behind the agents that use it. Active memory closes that gap, and the reason it can be done safely in a regulated enterprise is the same reason the rest of this platform is safe: the unverified thing is quarantined and must earn promotion. Provenance is the mechanism, and it is the same artifact a regulator wants, a record of who asserted what, with what authority, when.

MNEME also composes with the rest of the series rather than standing alone. Its nodes are RAILS nodes, so an ontology operation is a typed, deterministic step in a compiled graph. Its quarantine of low-confidence writes is the memory-side analogue of hallucination containment. Its provenance and pending-review queue are governable by the same policy engine that evidences every other agent action. Structured memory is not a separate product bolted to the platform; it is the layer that gives the agents a shared, durable, auditable model of the enterprise's world, and lets them improve it without being trusted blindly.

10 Limitations

- **The empirical evaluation covers retrieval, not write-back.** Section 8 measures the vector-seeded graph-hop mechanism alone, on a public multi-hop benchmark with a per-question mention graph; it does not measure promotion accuracy, false-promotion rate, or behavior against the real Postgres/Neo4j backends, and the graph-hop heuristic itself is a net negative on comparison-type questions.
- **Promotion policy is a judgment.** Reinforcement thresholds and confidence bars trade richness against caution; set them loose and the graph fills with weakly-corroborated concepts, set them tight and useful additions languish in staging.
- **Deduplication is similarity-bounded.** The pre-flight check catches near-duplicates by embedding similarity; a genuinely novel phrasing of an existing concept that is semantically distant can still slip through as a new entry until review catches it.
- **One-way PDM import.** The data model is the source of truth and the import is one-directional by design; the ontology does not write back to the PDM, so corrections discovered in the graph must flow to operational systems by a separate path.
- **Backend parity is behavioral, not identical.** Postgres and Neo4j are contract-tested for equivalent behavior, but deep multi-hop traversal is capped more tightly on Postgres; very graph-heavy ontologies belong on Neo4j.
- **Tag quality depends on chunking.** The default semantic Tag works best on short, entity-like text; long or discursive passages benefit from an LLM extraction step in front, which reintroduces a model call the pure-semantic path avoids.

11 Conclusion

Enterprises have accepted that agents need a knowledge graph, and have almost universally built it read-only, leaving the institutional memory unable to learn from the agents that depend on it. MNEME makes the ontology active memory instead: agents read it, validate against it, and write back to it, while provenance, deduplication, conflict detection, low-confidence staging, quarantine from canonical retrieval, and earned promotion ensure that a contribution improves the graph only when it has proven itself. The authoritative core, imported from the platform data model, stays read-only to agents throughout. The effect is a structured memory that gets richer every run with none of the regulatory blast radius that has kept everyone else read-only, a knowledge graph agents can write to, safely, which is the capability the next generation of enterprise agents will be built on.

References

[1] Gruber, T. R. *A Translation Approach to Portable Ontology Specifications.* Knowledge Acquisition, 5(2):199–220, 1993.

[2] Edge, D., Trinh, H., Cheng, N., et al. *From Local to Global: A Graph RAG Approach to Query-Focused Summarization.* Microsoft Research, 2024. arXiv:2404.16130. (Graph retrieval multi-hop accuracy vs vector baselines.)

[3] Lewis, P., Perez, E., Piktus, A., et al. *Retrieval-Augmented Generation for Knowledge-Intensive NLP Tasks.* Advances in Neural Information Processing Systems (NeurIPS) 33, 2020.

[4] World Wide Web Consortium. *PROV-O: The PROV Ontology.* W3C Recommendation, 2013.

[5] Hogan, A., Blomqvist, E., Cochez, M., et al. *Knowledge Graphs.* ACM Computing Surveys, 54(4):1–37, 2021.

[6] Gartner. *Top Trends in Data and Analytics, 2026 (Semantic Layers and GraphRAG).* Gartner Research, 2026.

[7] Neo4j. *Aura Agent: Ontology-Driven Agent Construction with Hosted MCP.* Neo4j product announcement, February 2026.

[8] *Memory in the Age of AI Agents.* arXiv:2512.13564, 2026.

[9] Reimers, N., & Gurevych, I. *Sentence-BERT: Sentence Embeddings using Siamese BERT-Networks.* Proc. EMNLP 2019. (MiniLM sentence-transformer family.)

[10] National Institute of Standards and Technology. *Artificial Intelligence Risk Management Framework (AI RMF 1.0).* NIST AI 100-1, 2023.

[11] European Parliament and Council. *Regulation (EU) 2024/1689 (Artificial Intelligence Act).* Official Journal of the European Union, 2024.

[12] International Organization for Standardization. *ISO/IEC 42001:2023, Information technology, Artificial intelligence, Management system.* ISO, 2023.

[13] Yang, Z., Qi, P., Zhang, S., Bengio, Y., Cohen, W. W., Salakhutdinov, R., & Manning, C. D. *HotpotQA: A Dataset for Diverse, Explainable Multi-hop Question Answering.* Proc. EMNLP 2018. arXiv:1809.09600.

Appendix A Node Output Keys

Each toolkit node writes well-known keys to the workflow state, suitable for prompt injection or downstream branching.

Node	Output keys
Retrieve	backend, items[], ontology_context (flat text for prompt injection)
Classify	concept_id, label, category, confidence, matched, backend
Describe	concept, ancestors[], grounding_text (formatted)
Expand	related[], ids[] (flat list)
Tag	tags[], tag_ids[] (deduplicated), backend
Deduplicate	recommended_action, is_duplicate, most_similar, neighbors[], backend
Validate	valid, violations[], backend (sets a halt signal on strict failure)
Memory Write	staged, conflict

Appendix B The Provenance and Conflict Model

The trust model that makes write-back safe, in one place.

Element	Values	Effect
Provenance	pdm_import / agent_write / manual	pdm_import is authoritative and read-only to agents; agent_write is staged and quarantined; manual is explicit human entry
Confidence	0–1; agent writes staged at ≤ 0.6	Below the bar a fact is durable but excluded from canonical retrieval
Conflict: label clash	same label, different concept	Logged before stage; routed for review
Conflict: category mismatch	proposed category contradicts structure	Logged before stage; routed for review
Conflict: relation contradicts	new relation contradicts an existing one	Logged before stage; never silently overwrites
Promotion	reinforcement / human review / API	Moves a staged fact to canonical; validation against IS_A inheritance must pass
Validation strictness	strict / warn / advisory	Strict halts the flow on a schema violation

Appendix C PDM-to-Ontology Mapping

The one-way import that bootstraps an ontology from an existing platform data model without duplicating it. Re-import preserves agent extensions and updates only PDM-sourced fields.

Platform Data Model	Becomes in the ontology
Entity	Concept (provenance = pdm_import, read-only to agents)
Foreign-key relationship	Relation between concepts
Type / subtype hierarchy	IS_A relation (drives inherited-attribute validation)
Attribute / column	Concept attribute in the schema
(agent-added structure)	Preserved across re-import; only PDM-sourced fields refreshed

FlowX.AI

FlowX.AI builds and orchestrates enterprise-grade AI agents for regulated industries, with a focus on banking, financial services, and insurance. The FlowX.AI Ontology Layer gives agents a shared, provenance-tracked structured memory they can read, validate against, and safely contribute to, so institutional knowledge stays consistent and grows with use rather than decaying between manual governance cycles.

FlowX.AI Technical Paper · MNEME v1.1 · Bogdan Răduță, Head of Research · bogdan.raduta@flowx.ai