ByteRover: Agent-Native Memory Through LLM-Curated Hierarchical Context

Andy Nguyen[1], Danh Doan[1], Hoang Pham[1], Bao Ha[1], Dat Pham[1], Linh Nguyen[1], Hieu Nguyen[1], Thien Nguyen[1], Cuong Do[1], Phat Nguyen[1], Toan Nguyen[1]

[1]ByteRover
https://www.byterover.dev

Abstract

Memory-Augmented Generation (MAG) extends large language models with external memory to support long-context reasoning, but existing approaches universally treat memory as an *external service* that agents call into—delegating storage to separate pipelines of chunking, embedding, and graph extraction. This architectural separation means the system that stores knowledge does not understand it, leading to semantic drift between what the agent intended to remember and what the pipeline actually captured, loss of coordination context across agents, and fragile recovery after failures. In this paper, we propose ByteRover, an *agent-native* memory architecture that inverts the memory pipeline: the same LLM that reasons about a task also curates, structures, and retrieves knowledge. ByteRover represents knowledge in a hierarchical *Context Tree*—a file-based knowledge graph organized as *Domain* > *Topic* > *Subtopic* > *Entry*—where each entry carries explicit relations, provenance, and an Adaptive Knowledge Lifecycle (AKL) with importance scoring, maturity tiers, and recency decay. Retrieval uses a 5-tier progressive strategy that resolves most queries at sub-100 ms latency without LLM calls, escalating to agentic reasoning only for novel questions. Experiments on LoCoMo and LongMemEval demonstrate that ByteRover achieves state-of-the-art accuracy on LoCoMo and competitive results on LongMemEval while requiring *zero external infrastructure*—no vector database, no graph database, no embedding service—with all knowledge stored as human-readable markdown files on the local filesystem.

1 Introduction

Large Language Models (LLMs) have demonstrated remarkable capabilities across a wide range of tasks [Brown et al., 2020, Achiam et al., 2023, Wei et al., 2022], yet they remain fundamentally limited in their ability to maintain and reason over long-term context. These models process information within a finite attention window, and their internal representations do not persist across interactions, causing earlier details to be forgotten once they fall outside the active context [Brown et al., 2020, Beltagy et al., 2020]. Even within a single long sequence, attention effectiveness degrades with distance due to *attention dilution*, positional encoding limitations, and token interference, leading to the well-known "lost-in-the-middle" phenomenon [Liu et al., 2024, Press et al., 2022].

To address these limitations, Memory-Augmented Generation (MAG) systems have emerged as a promising direction for enabling LLMs to operate beyond the boundaries of their fixed context windows [Xu et al., 2025, Nan et al., 2025, Jiang et al., 2026a, Chhikara et al., 2025]. MAG equips an agent with an external memory module that continuously records interaction histories and allows the agent to retrieve and reintegrate past experiences when generating new responses. The paradigm

has rapidly evolved from lightweight semantic stores to entity-centric, episodic, and hierarchical designs [Jiang et al., 2026b].

Despite this architectural diversity, current MAG systems share a common structural pattern: **memory is an external service that agents call into**. The agent serializes data, sends it to a separate pipeline (for chunking, embedding, entity extraction, or graph construction), receives an acknowledgment, and later queries the service to retrieve results. The pipeline that stores knowledge does not understand it—chunking is mechanical, embeddings encode surface similarity rather than semantic relationships, and the agent has no visibility into how its memories were organized or why certain relationships were created.

This external-service paradigm creates three failure modes that become critical for autonomous agents operating over long time horizons:

1. **Semantic drift.** The agent's understanding of what it stored diverges from what the memory service actually captured. The agent intends to store a nuanced insight; the pipeline chunks and embeds it differently, and the next retrieval returns a tangentially related fragment.

2. **Lost coordination context.** When multiple agents share an external memory service, they share data but not understanding. Agent A stores a finding with reasoning and rationale. Agent B retrieves the data but lacks the *why*—what reasoning led to the conclusion, what actions were expected to follow. The provenance is lost in the embedding.

3. **Recovery fragility.** When an autonomous agent crashes mid-task, it must reconstruct state by querying the memory service, interpreting results, and inferring where it left off. With stateful, file-based memory, the state is *in the files*—per-operation status, timestamps, and the knowledge structure itself tells the agent exactly what was completed.

To address these limitations, we propose ByteRover, an *agent-native* memory architecture that inverts the relationship between agent and memory. Instead of calling an external memory service, the same LLM that reasons about a task also curates knowledge into a hierarchical *Context Tree*—a file-based knowledge graph where each entry carries explicit relations, provenance, and lifecycle metadata. Memory operations (ADD, UPDATE, UPSERT, MERGE, DELETE) are tools in the agent's toolkit, not API calls to an external service, enabling a stateful feedback loop where the agent sees per-operation results and adapts in real time.

Our contributions are summarized as follows:

1. We propose ByteRover, an agent-native memory architecture where the LLM itself curates knowledge through structured operations with explicit provenance and rationale, eliminating the separation between the "understanding" agent and the "storing" pipeline.

2. We introduce the *Context Tree*, a hierarchical file-based knowledge graph with an Adaptive Knowledge Lifecycle (AKL)—importance scoring, maturity tiers (draft $\rightarrow$ validated $\rightarrow$ core), and recency decay—that enables knowledge to naturally evolve over time.

3. We design a 5-tier progressive retrieval strategy that resolves most queries at sub-100 ms latency without LLM calls, combined with out-of-domain detection that explicitly signals when queries fall outside stored knowledge.

4. We demonstrate that ByteRover achieves state-of-the-art results on LoCoMo and competitive results on LongMemEval benchmarks while requiring zero external infrastructure—no vector database, no graph database, no embedding service—with all knowledge stored as human-readable markdown files.

2 Background

2.1 Memory-Augmented Generation

Agentic memory extends retrieval-based generation by introducing persistent, writable memory that evolves across interactions [Jiang et al., 2026b]. Formally, at step t, the agent conditions on observations o_t and an external memory state $\mathcal{M}_t$:

$$y_t \sim f_\theta\Big(\phi(o_t, s_t) \ \oplus \ \psi(\mathcal{M}_t; q_t)\Big), \tag{1}$$

where y_t denotes the output, s_t is additional agent state, $\psi(\mathcal{M}_t; q_t)$ retrieves memory given query q_t, and $\oplus$ represents integration (e.g., prompt concatenation). Crucially, memory affects behavior through the explicit retrieval term ψ rather than updates to θ.

Two coupled processes are operated: **inference-time recall** (reading memory to condition decisions) and **memory update** (writing, consolidating, and forgetting to maintain a useful long-term store). The memory module evolves via a feedback loop:

$$o_t = \mathrm{LLM}(q_t, \mathrm{Retrieve}(q_t, \mathcal{M}_t)), \tag{2}$$

$$\mathcal{M}_{t+1} = \mathrm{Update}(\mathcal{M}_t, q_t, o_t). \tag{3}$$

2.2 Taxonomy of Existing Approaches

Following the taxonomy introduced by Jiang et al. [2026b], existing MAG systems can be organized into four structural categories:

- **Lightweight Semantic Memory.** Independent textual units embedded in a vector space and retrieved via top-k similarity. No explicit structural relations [Liu et al., 2026].
- **Entity-Centric and Personalized Memory.** Organizes information around explicit entities using structured records or attribute-value pairs [Xu et al., 2025, Modarressi et al., 2023, Chhikara et al., 2025].
- **Episodic and Reflective Memory.** Adds temporal abstraction by organizing interactions into episodes or higher-level summaries [Nan et al., 2025].
- **Structured and Hierarchical Memory.** Imposes explicit organization over stored information via graphs [Jiang et al., 2026a, Rasmussen et al., 2025], hierarchical tiers [Kang et al., 2025a, Packer et al., 2023], or policy-optimized management.

2.3 The External Service Paradigm

Despite their architectural diversity, *all* systems in the taxonomy above share a common interaction pattern: the agent communicates with memory through an API boundary. The agent serializes data, the memory service processes it through its own pipeline (chunking, embedding, entity extraction, graph construction), and the agent later queries the service to retrieve results.

This pattern has a fundamental consequence: **the system that stores knowledge does not understand it.** The embedding model that creates vector representations operates independently of the agent's reasoning. The entity extraction pipeline has its own notion of what matters. The agent has no visibility into how its memories were organized, why certain relationships were created, or whether the stored representation faithfully captures its intent.

ByteRover departs from this paradigm by making the agent itself the curator—the same LLM that reasons about a task decides what to store, where to place it, what it relates to, and why it matters.

3 The *Context Tree*

This section introduces the ByteRover architecture and its core data structure, the *Context Tree*.

3.1 Architectural Overview

ByteRover is organized into three logical layers, illustrated in Figure 1:

- **Agent Layer.** The LLM reasoning loop that produces both task outputs and memory operations. Memory tools (curate, query, search) are available as first-class tools alongside file I/O, code execution, and other agent capabilities.
- **Execution Layer.** A sequential task queue that processes curate and query operations. Curation runs through a sandboxed environment where the LLM's generated code has controlled access to the knowledge layer via a `ToolsSDK` interface. The sequential queue eliminates write-write conflicts without file-level locking.
- **Knowledge Layer.** The *Context Tree* (a hierarchical file structure of markdown entries), the MiniSearch full-text index, and the query cache. All storage is local filesystem—no external databases or services.

The key design principle is that memory operations are *tools in the agent's toolkit*, not API calls to an external service. When the agent curates knowledge, it produces structured operations that execute within the agent process, with per-operation feedback enabling real-time error recovery. When the agent queries knowledge, results are returned from in-process caches and indexes before any LLM call is considered. Each project is served by a single agent process with its own *Context Tree*, managed by a per-project agent pool. When multiple clients (TUI, CLI, MCP) submit tasks to the same project, a sequential, deduplicated task queue serializes all operations, eliminating write-write conflicts without file-level locking. ByteRover exposes `brv-query` and `brv-curate` as MCP tools, enabling integration with any MCP-compatible agent framework. All knowledge is stored as human-readable markdown files on the local filesystem—version-controllable, portable, and requiring zero external infrastructure.

3.2 The Context Tree: Data Structure

The *Context Tree* is a hierarchical file-based knowledge graph organized as *Domain* > *Topic* > *Subtopic* > *Entry*. We formalize it as a directed graph $\mathcal{G} = (\mathcal{N}, \mathcal{E})$ where nodes $\mathcal{N}$ are knowledge entries (markdown files) and edges $\mathcal{E}$ are explicit cross-references declared via `@domain/topic/file.md` relation annotations.

3.2.1 Knowledge Entry Structure

Each entry $n_i \in \mathcal{N}$ is a standalone markdown file with structured content (Equation 4, Appendix C):

$$n_i = \langle \mathcal{R}_i, \mathcal{C}_i, \mathcal{V}_i, \mathcal{S}_i, \mathcal{L}_i \rangle, \tag{4}$$

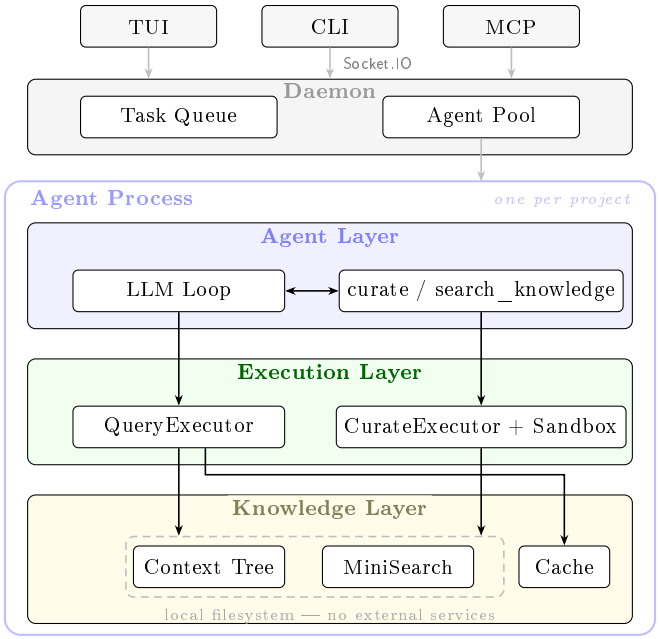

Figure 1: Architectural overview of ByteRover. Clients (TUI, CLI, MCP) connect via Socket.IO to a daemon that manages a per-project task queue and agent pool. Each agent process contains three logical layers: (1) an *Agent Layer* where `curate` and `search_knowledge` are first-class tools in the LLM's reasoning loop; (2) an *Execution Layer* with a query executor for 5-tier progressive retrieval and a sandboxed curation environment; and (3) a *Knowledge Layer* with the *Context Tree*, BM25 full-text index, and query cache, all backed by the local filesystem with no external infrastructure.

where $\mathcal{R}_i$ denotes the relation set (explicit edges to other entries), $\mathcal{C}_i$ is the raw concept (provenance: task, changes, sources, timestamp, author), $\mathcal{V}_i$ is the narrative (interpreted structure: dependencies, rules, examples, diagrams), $\mathcal{S}_i$ contains snippets (code, formulas, raw data), and $\mathcal{L}_i$ is the lifecycle metadata.

3.2.2 Relation Graph and Symbol Tree

The edge set $\mathcal{E}$ is constructed from explicit `@relation` annotations in the `Relations` section of each entry. Unlike embedding-based implicit similarity, these edges represent *author-stated* semantic connections—the LLM that created the entry decided that these concepts are related and stated why.

A bidirectional reference index maintains both forward links (source $\rightarrow$ targets it references) and backlinks (target $\rightarrow$ sources that reference it), enabling graph traversal in both directions with O(1) lookup per entry.

A hierarchical *symbol tree* provides O(1) lookup from relative paths to knowledge entries and hosts the reference index above. The tree supports five symbol kinds: *Domain* (1), *Topic* (2), *Subtopic* (3), *Context* (4), and *Summary* (5). For query and curate operations, a lightweight representation of the tree structure is injected into the agent's system prompt: either a directory listing of domain and topic names (up to 200 entries) or, when full-text search is available, a compact instruction to use the search tool. This gives the agent ambient awareness of what knowledge exists without dumping full contents.

3.2.3 Adaptive Knowledge Lifecycle (AKL)

Each entry carries lifecycle metadata $\mathcal{L}_i$ that governs its evolution over time through an *Adaptive Knowledge Lifecycle* (AKL) mechanism:

- **Importance score** $\iota_i \in [0, 100]$: Tracks the value of each entry over time. Access events contribute a +3 bonus; update events contribute +5. A daily decay factor of 0.995 prevents unbounded accumulation.
- **Maturity tiers**: Entries progress through three tiers based on importance, with hysteresis gaps to prevent rapid oscillation: *draft* $\rightarrow$ *validated* (promotion at $\iota \geq 65$, demotion at $\iota < 35$; gap of 30), *validated* $\rightarrow$ *core* (promotion at $\iota \geq 85$, demotion at $\iota < 60$; gap of 25).
- **Recency decay**: A time-dependent score $r_i = \exp(-\Delta t_i / \tau)$ where Δt_i is the number of days since last update and $\tau = 30$ is the decay constant ($\sim$21-day half-life).

The compound retrieval score (Equation 5) combines search relevance with lifecycle signals:

$$\text{Score}(n_i, q) = w_r \cdot \text{BM25}(n_i, q) + w_\iota \cdot \hat{\iota}_i + w_t \cdot r_i, \tag{5}$$

where $\hat{\iota}_i$ is the normalized importance and w_r, w_ι, w_t are tunable weights.

4 Agent-Native Operations

4.1 LLM-Curated Knowledge Operations

ByteRover supports five atomic curate operations (Table 1) that the LLM can compose:

Operation	Behavior
`ADD`	Create new entry; auto-generate `context.md` at each hierarchy level
`UPDATE`	Replace content of an existing entry
`UPSERT`	Add if new, update if exists (reduces pre-check overhead)
`MERGE`	Combine two entries intelligently; delete the source
`DELETE`	Remove a single entry or an entire subtree

Table 1: The five atomic curate operations. Every operation carries a `reason` field that serves as an audit trail.

4.1.1 Curation Pipeline

Curation follows a three-phase process:

1. **Preprocessing.** Source documents are read and validated (max 5 files, 40K characters each). PDFs are converted to text; code files are truncated to 2000 lines.
2. **Pre-Compaction.** An escalated compression strategy reduces input size through three levels: (L1) LLM summarization, (L2) aggressive LLM summarization at 0.6$\times$ token budget, (L3) deterministic binary-search prefix truncation (guaranteed convergence). This ensures curation always terminates regardless of input size.

Tier	Mechanism	Latency	Condition
0	Exact cache hit	$\sim$0 ms	Hash match + valid fingerprint
1	Fuzzy cache (Jaccard)	$\sim$50 ms	Jaccard $\geq \theta_{\text{fuzzy}}$
2	Direct MiniSearch	$\sim$100 ms	BM25 score $\geq \theta_{\text{high}}$, sufficient gap
3	Optimized LLM call	$<$5 s	BM25 score $\geq \theta_{\text{med}}$
4	Full agentic loop	8–15 s	All other queries

Table 2: The five retrieval tiers with their latency characteristics and escalation conditions.

3. **Curation.** The LLM agent runs in a sandboxed environment with access to the `ToolsSDK`—a controlled interface providing `curate()`, `searchKnowledge()`, `readFile()`, and other file operations. The agent reads sources, reasons about patterns and relationships, and produces structured curate operations with explicit provenance.

4.1.2 Stateful Feedback Loop

A critical differentiator from external services is the *stateful feedback loop*. Each curate call returns per-operation status:

```
{
  "applied": [
    {"type": "UPSERT", "path": "analysis/semi", "status": "success"},
    {"type": "MERGE",  "path": "analysis/energy", "status": "failed",
     "message": "Source file not found"}
  ],
  "summary": {"added": 0, "deleted": 0, "updated": 1, "merged": 0, "failed": 1}
}
```

The agent sees which operations succeeded, which failed, and why. It can reason about failures and adapt—skip the operation, retry with corrections, or flag the gap for later resolution. This feedback loop is impossible when memory is an external service returning HTTP status codes.

4.1.3 Atomic Writes and Crash Safety

All file operations use an atomic write-to-temp-then-rename pattern. If the process crashes mid-write, the *Context Tree* remains consistent—no partial entries or corrupted knowledge.

4.2 5-Tier Progressive Retrieval

Retrieval uses a tiered strategy (Table 2) that minimizes LLM calls, illustrated in Figure 2 and formalized in Algorithm 1.

Tiers 0–2 resolve queries without any LLM call, returning cached or high-confidence search results directly. Tier 3 pre-fetches relevant documents and passes them to a single optimized LLM call with constrained output (1,024 tokens, temperature 0.3). Tier 4 is the full agentic fallback: the agent enters a multi-turn reasoning loop where it calls tools (`code_exec`, `readFile`) to navigate the *Context Tree*, with a higher token budget (2,048 tokens, temperature 0.5) and up to 50 iterations.

4.2.1 Search Engine

The search engine is MiniSearch—a lightweight full-text search library with BM25 ranking, fuzzy matching (0.2 character similarity threshold), and prefix search. Field boosting weights titles at 5×

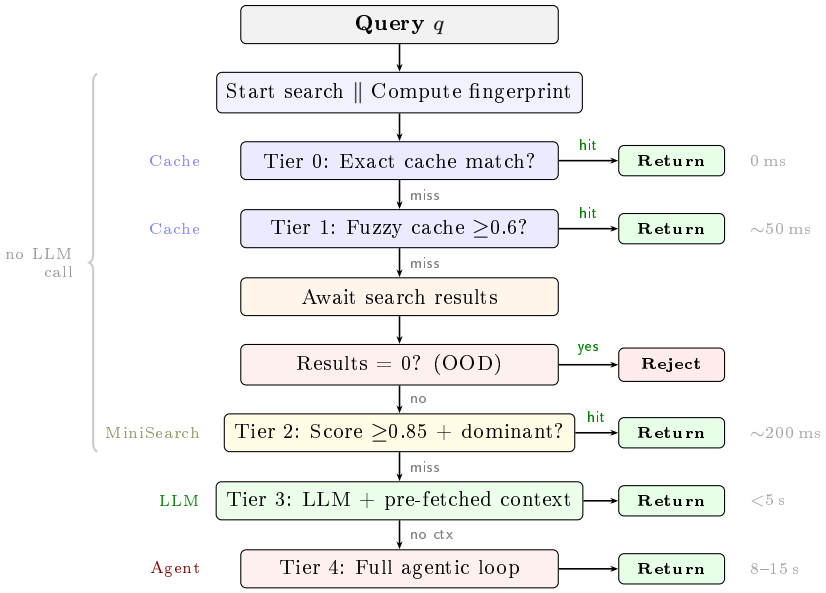

Figure 2: The 5-tier progressive retrieval pipeline. Search is initiated in parallel with fingerprint computation. Tiers 0–1 resolve from cache without awaiting search. Tier 2 serves high-confidence results directly from MiniSearch. Only novel or ambiguous queries escalate to Tier 3 (single optimized LLM call with pre-fetched context) or Tier 4 (full agentic loop with tool access). Approximate latencies shown on right.

and paths at 1.5× over content.

Score normalization (Equation 6) maps raw BM25 scores to $[0, 1)$ via:

$$\hat{s} = \frac{s_{\text{raw}}}{1 + s_{\text{raw}}}, \tag{6}$$

yielding interpretable thresholds: strong (15) → 0.94, medium (8) → 0.89, moderate (4) → 0.80, weak (1) → 0.50.

4.2.2 Out-of-Domain Detection

When significant query terms (length ≥ 4 characters) do not match any entry in the knowledge base and the normalized score falls below a threshold ($\theta_{\text{OOD}} = 0.85$), the system explicitly signals *"this query appears outside the scope of stored knowledge."* This prevents hallucinated answers from tangential results—an essential property when agents are making decisions based on retrieved knowledge.

5 Experiments

We conduct comprehensive experiments to evaluate both the reasoning effectiveness and system properties of ByteRover over state-of-the-art baselines (Tables 3–5).

5.1 Experimental Setup

Datasets. We evaluate on two widely adopted long-term conversational benchmarks: **(1) LoCoMo** [Maharana et al., 2024], which contains ultra-long conversations (average length of ∼20K to-

Algorithm 1 5-Tier Progressive Retrieval

Require: Query q, Cache $\mathcal{C}$, Search Index $\mathcal{I}$, Agent $\mathcal{A}$
Ensure: Response R, Tier τ

$h \leftarrow \text{Hash}(q)$
if $h \in \mathcal{C}$ **and** $\text{Fingerprint}(\mathcal{C}[h]) = \text{Fingerprint}(\mathcal{G})$ **then**
  return $\mathcal{C}[h]$, $\tau = 0$ {Exact cache hit}
end if
$q' \leftarrow \arg\max_{c \in \mathcal{C}} \text{Jaccard}(q, c)$
if $\text{Jaccard}(q, q') \geq \theta_{\text{fuzzy}}$ **then**
  return $\mathcal{C}[q']$, $\tau = 1$ {Fuzzy cache hit}
end if
$\mathcal{D} \leftarrow \text{MiniSearch}(\mathcal{I}, q)$ {BM25 + fuzzy + prefix}
if $\text{Score}(\mathcal{D}_1) \geq \theta_{\text{high}}$ **and** $\text{Gap}(\mathcal{D}_1, \mathcal{D}_2) \geq \theta_{\text{gap}}$ **then**
  return $\text{DirectResponse}(\mathcal{D})$, $\tau = 2$ {High-confidence search}
end if
if $\text{Score}(\mathcal{D}_1) \geq \theta_{\text{med}}$ **then**
  $C_{\text{pre}} \leftarrow \text{Prefetch}(\mathcal{D})$
  return $\text{LLM}(q, C_{\text{pre}})$, $\tau = 3$ {Optimized single LLM call}
end if
return $\mathcal{A}.\text{AgenticLoop}(q)$, $\tau = 4$ {Full multi-turn reasoning}

kens across 35 sessions) designed to assess long-range temporal and causal retrieval; and **(2) LongMemEval-S** [Wu et al., 2025], the full-scale variant of LongMemEval with 500 questions spanning six memory-ability categories, average context length exceeding 100K tokens across ∼48 sessions per question, designed to stress-test memory retention and scalability under realistic interaction horizons.

Baselines. On LoCoMo, we evaluate six memory systems under a unified protocol using our benchmark harness with identical judge configuration, enabling direct comparison: **Mem0** [Chhikara et al., 2025], **Zep** [Rasmussen et al., 2025], **Hindsight** [Latimer et al., 2025], **HonCho**,[1] **Memobase**,[2] and **OpenAI Memory** (ChatGPT). These systems span the spectrum from structured fact extraction (Mem0, Memobase) to graph-based memory architectures (Zep, Hindsight) and commercial assistants (OpenAI Memory, HonCho).

On LongMemEval-S, we include our own evaluations of Hindsight and HonCho alongside published results from **Chronos** [Sen et al., 2026], **SmartSearch** [Derehag et al., 2026], **Memora** [Xia et al., 2026], **TiMem** [Li et al., 2026], Zep, and a full-context baseline, cited from their respective publications. Results marked with † use different backbone and judge configurations; see Table 4 for details.

Metrics. We adopt LLM-as-a-Judge [Zheng et al., 2023] as the primary metric, where a judge model evaluates each generated answer against the ground truth and returns a binary correctness label. For our own evaluations, we use Gemini 3 Flash as the judge with a separate justifier model (Gemini 3.1 Pro) that synthesizes an answer from retrieved context before scoring. Both models run at temperature 0.0. The judge uses a token budget of 8,192 with thinking disabled, while the justifier uses 32,768 tokens with low thinking budget. Full hyperparameter details are provided in

[1] `https://honcho.dev`
[2] `https://github.com/memodb-io/memobase`

Method	Single-Hop	Multi-Hop	Open-Domain	Temporal	Overall
HonCho	93.2	84.0	77.1	88.2	89.9
Hindsight	86.2	70.8	**95.1**	83.8	89.6
Memobase	70.9	46.9	77.2	85.1	75.8
Zep	74.1	66.0	67.7	79.8	75.1
Mem0	67.1	51.2	72.9	55.5	66.9
OpenAI Memory	63.8	42.9	62.3	21.7	52.9
ByteRover (ours)	**97.5**	**93.3**	85.9	**97.8**	**96.1**

Table 3: LLM-as-Judge accuracy (%) on LoCoMo (4 categories, 1,982 questions). All systems evaluated under identical conditions using our benchmark harness with Gemini 3 Flash as the judge.

Appendix B.

Evaluation prompts. To enable direct, apples-to-apples comparison with Hindsight [Latimer et al., 2025], we reuse their publicly available evaluation prompts.[3] Specifically, the judge prompts—including the default LoCoMo preamble and all five LongMemEval category-specific preambles (standard, temporal-reasoning, knowledge-update, and preference)—are adopted without modification, ensuring that scoring criteria are identical across systems. The LoCoMo justifier prompt is likewise reused verbatim. For the LongMemEval justifier, we adapt Hindsight's prompt to reflect ByteRover's single-layer knowledge representation: where Hindsight instructs the model to cross-reference atomic facts against raw source chunks (a two-layer retrieval output), our variant replaces this with guidance for interpreting *key facts* with session-level timestamps, matching the *Context Tree*'s curated entry format. All other instructions—date calculation rules, counting-question disambiguation, recommendation handling, and question-type-specific guidelines—remain unchanged from the original.

5.2 Overall Comparison on LoCoMo

ByteRover achieves the highest overall accuracy at 96.1%, outperforming the next-best system (HonCho, 89.9%) by 6.2 percentage points. The gains are particularly pronounced on *multi-hop* questions (+9.3 over HonCho), which require synthesizing information across distant sessions—a task where the *Context Tree*'s explicit inter-entry relations provide navigable paths that flat memory stores lack. On *temporal* queries, ByteRover reaches 97.8%, benefiting from structured timestamps embedded in each entry's metadata that enable precise temporal grounding. The sole category where ByteRover does not lead is *open-domain*, where Hindsight achieves 95.1% versus 85.9%. Open-domain questions frequently require commonsense reasoning that extends beyond the conversation corpus, an area where retrieval-augmented approaches can excel by leveraging the backbone LLM's parametric knowledge.

5.3 Generalization on LongMemEval-S

ByteRover achieves the highest overall accuracy at 92.8%, surpassing Chronos-Low (92.6%, GPT-4o backbone) while remaining below Chronos-High (95.6%, Claude Opus 4.6 backbone), which benefits from a substantially more capable generation model. The strength profile is distinctive:

[3] `https://github.com/vectorize-io/hindsight/tree/main/hindsight-dev/benchmarks`

Method	KU	SSU	SSA	SSP	TR	MS	Overall
Chronos†	96.2	94.3	**100.0**	80.0	90.2	**91.7**	92.6
Hindsight	94.9	97.1	96.4	80.0	91.0	87.2	91.4
HonCho	94.9	94.3	96.4	90.0	88.7	85.0	90.4
SmartSearch†	93.6	**100.0**	85.7	**96.7**	82.7	84.2	88.4
Memora†	97.4	98.6	78.6	83.3	89.5	78.2	87.4
TiMem†	87.7	96.3	85.7	55.3	73.4	72.8	79.0
Zep†	83.3	92.9	80.4	56.7	62.4	57.9	71.2
Full-context†	78.2	81.4	94.6	20.0	45.1	44.3	60.2
ByteRover (ours)	**98.7**	98.6	98.2	**96.7**	**91.7**	84.2	**92.8**

Table 4: LLM-as-Judge accuracy (%) on LongMemEval-S (500 questions). KU = Knowledge Update, SSU = Single-Session User, SSA = Single-Session Assistant, SSP = Single-Session Preference, TR = Temporal Reasoning, MS = Multi-Session. Hindsight and HonCho evaluated with our harness (Gemini 3 Flash judge). †Results from respective papers; backbone and judge configurations differ.

Metric	LoCoMo	LongMemEval-S
Context tree size	272 docs	23,867 docs
Total queries	1,982	500
Cold query latency p50	1.2 s	1.6 s
Cold query latency p95	1.4 s	2.3 s
Cold query latency p99	1.7 s	2.5 s

Table 5: Operational latency profile. Cold query latency measures end-to-end time from process invocation through response delivery, excluding answer justification and evaluation.

ByteRover leads on knowledge update (98.7%), temporal reasoning (91.7%), and single-session preference (96.7%, tied with SmartSearch), categories that reward precise fact retrieval and temporal grounding—capabilities directly served by the *Context Tree*'s structured entries and timestamp-aware indexing. The weakest category is multi-session (84.2%), which requires cross-session synthesis over long horizons and represents the primary area for improvement. Notably, Chronos achieves 91.7% on multi-session, suggesting that its event-ordering approach may better capture inter-session dependencies.

5.4 Operational Profile

Despite a substantially larger context tree on LongMemEval-S (23,867 documents vs. 272 on LoCoMo), median query latency remains low at 1.6 s, suggesting that the tiered retrieval architecture effectively bounds search cost as the corpus grows. The tight percentile spread (p50-to-p99 gap of 0.5 s on LoCoMo, 0.9 s on LongMemEval-S) indicates consistent performance without long-tail degradation.

5.5 Ablation Study

To assess the contribution of individual components, we conduct an ablation study on LongMemEval-S by selectively disabling three key query-time mechanisms. Ablation experiments are conducted on LongMemEval-S, where the larger corpus (23,867 documents) provides a more demanding test

Configuration	KU	SSU	SSA	SSP	TR	MS	Overall	Δ
w/o Tiered Retrieval	69.2	71.4	76.8	83.3	61.7	47.4	63.4	−29.4
w/o OOD Detection	98.7	98.6	98.2	96.7	89.5	85.0	92.4	−0.4
w/o Relation Graph	98.7	98.6	98.2	96.7	89.5	85.0	92.4	−0.4
ByteRover (Full)	98.7	98.6	98.2	96.7	91.7	84.2	**92.8**	—

Table 6: Ablation study on LongMemEval-S (500 questions). Each row disables one query-time component while keeping the curated *Context Tree* unchanged. Category abbreviations follow Table 4.

of retrieval-side components; LoCoMo results (Table 3) serve as a generalization test on a smaller corpus.

w/o Tiered Retrieval. All queries are routed to the full agentic loop (Tier 4), bypassing Tiers 0–3 and the separate justifier model (see §4.2 for the distinction between tiers). This ablation produces the largest accuracy drop (−29.4 pp, from 92.8% to 63.4%). Multi-session questions suffer most severely (84.2% → 47.4%), followed by temporal reasoning (91.7% → 61.7%). The result demonstrates that the tiered architecture is not merely a latency optimization: the lower tiers surface precise, high-confidence content that the justifier can faithfully synthesize, whereas unconstrained agentic reasoning over a 23,867-document corpus compounds retrieval errors with generation errors.

w/o OOD Detection. Disabling the out-of-domain rejection gate produces only a modest overall decline (−0.4 pp, from 92.8% to 92.4%), with the effect concentrated in temporal reasoning (91.7% → 89.5%, −2.2 pp). Four categories (KU, SSU, SSA, SSP) are completely unaffected, indicating that OOD detection primarily guards against temporal and multi-session queries where partial matches from unrelated sessions could otherwise mislead the justifier. The small magnitude suggests the curated *Context Tree* already provides strong topical coherence, limiting the opportunities for OOD errors.

w/o Relation Graph. Removing explicit cross-entry relations produces the same overall decline as disabling OOD detection (−0.4 pp, 92.4%), with an identical per-category profile: temporal reasoning drops by 2.2 pp (91.7% → 89.5%) while multi-session improves slightly (84.2% → 85.0%). The identical scores to the OOD ablation suggest that, on LongMemEval-S's question distribution, relation edges and OOD gates address overlapping failure modes—both primarily affect queries that require disambiguating temporally adjacent sessions. The relation graph's contribution may be more pronounced on benchmarks with explicit multi-hop reasoning demands (e.g., LoCoMo's multi-hop category, where relations provide navigable paths across conversation boundaries).

Components not ablated. Three mechanisms—Adaptive Knowledge Lifecycle (AKL), the curation feedback loop, and escalated compression—operate exclusively during the write path (curation). Since the curated *Context Tree* is held constant across all ablation conditions, their contribution cannot be isolated through query-time ablation on a static benchmark. Evaluating these components would require re-curation under degraded conditions, which we leave to future work.

6 Conclusion

We introduced ByteRover, an agent-native memory architecture that inverts the conventional MAG paradigm: instead of delegating knowledge storage to an external service, the same LLM that reasons about a task curates knowledge into a hierarchical *Context Tree* with explicit relations, provenance, and lifecycle management. By making memory operations first-class tools in the agent's reasoning loop, ByteRover enables a stateful feedback loop that eliminates semantic drift, preserves coordination context across agents, and supports graceful recovery from failures.

The 5-tier progressive retrieval strategy resolves most queries at sub-100 ms latency without LLM calls, while the Adaptive Knowledge Lifecycle enables knowledge to naturally evolve—reinforcing frequently accessed entries and decaying stale ones. Empirical results on LoCoMo and LongMemEval demonstrate that ByteRover achieves competitive or state-of-the-art accuracy while requiring zero external infrastructure, with all knowledge stored as human-readable markdown files on the local filesystem.

7 Limitations

While ByteRover demonstrates strong architectural properties, it has several limitations that warrant discussion.

First, the **write path is expensive**. LLM-curated knowledge requires reasoning per curation event, which is slower and more costly than mechanical chunking and embedding. For use cases where write throughput is critical (e.g., real-time ingestion of high-frequency data streams), the curation overhead may be prohibitive.

Second, **novel queries are slower than vector search**. When queries miss the cache and index (Tier 3–4), ByteRover requires an LLM call that a vector similarity search does not. The design assumes that agents ask many variations of a smaller set of questions, so the cache absorbs most of the load. When this assumption does not hold, retrieval latency increases.

Third, **curation quality depends on backbone model capability**. This is a shared limitation across all MAG systems [Jiang et al., 2026b], but ByteRover's deeper reliance on LLM reasoning for both storage and retrieval amplifies the impact of backbone sensitivity. Open-weight models with higher format error rates may produce lower-quality knowledge entries.

Fourth, the **file-based storage may face scaling challenges** at very large knowledge bases. The in-memory MiniSearch index and sequential task queue are designed for knowledge bases of up to $\sim$10K entries. Beyond this scale, sharding strategies or alternative indexing backends may be needed.

Finally, the **sequential task queue limits write throughput** under heavy concurrent load. While this is rarely a bottleneck in practice (curation is an infrequent, high-cost operation), deployments with many agents writing simultaneously may experience queuing delays.

References

Josh Achiam, Steven Adler, Sandhini Agarwal, Lama Ahmad, Ilge Akkaya, Florencia Leoni Aleman, Diogo Almeida, Janko Altenschmidt, Sam Altman, Shyamal Anadkat, et al. Gpt-4 technical report. *arXiv preprint arXiv:2303.08774*, 2023.

Iz Beltagy, Matthew E Peters, and Arman Cohan. Longformer: The long-document transformer. *arXiv preprint arXiv:2004.05150*, 2020.

Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. Language models are few-shot learners. *Advances in Neural Information Processing Systems*, 33:1877–1901, 2020.

Prateek Chhikara, Dev Khant, Saket Aryan, Taranjeet Singh, and Deshraj Yadav. Mem0: Building production-ready ai agents with scalable long-term memory. *arXiv preprint arXiv:2504.19413*, 2025.

Jesper Derehag, Carlos Calva, and Timmy Ghiurau. Smartsearch: How ranking beats structure for conversational memory retrieval. *arXiv preprint arXiv:2603.15599*, 2026.

Bernal Jiménez Gutiérrez, Yiheng Shu, Sizhe Qi, Weijian Zhou, and Yu Su. From rag to memory: Non-parametric continual learning for large language models. In *International Conference on Machine Learning*, 2025.

Chuanrui Hu, Xingze Gao, Zuyi Zhou, Dannong Xu, Yi Bai, Xintong Li, Hui Zhang, Tong Li, Chong Zhang, Lidong Bing, et al. Evermemos: A self-organizing memory operating system for structured long-horizon reasoning. *arXiv preprint arXiv:2601.02163*, 2026.

Dongming Jiang, Yi Li, Guanpeng Li, and Bingzhe Li. Magma: A multi-graph based agentic memory architecture for ai agents. *arXiv preprint arXiv:2601.03236*, 2026a.

Dongming Jiang, Yi Li, Songtao Wei, Jinxin Yang, Ayushi Kishore, Alysa Zhao, Dingyi Kang, Xu Hu, Feng Chen, Qiannan Li, and Bingzhe Li. Anatomy of agentic memory: Taxonomy and empirical analysis of evaluation and system limitations. *arXiv preprint arXiv:2602.19320*, 2026b.

Ziyan Jiang, Xueguang Ma, and Wenhu Chen. Longrag: Enhancing retrieval-augmented generation with long-context llms. *arXiv preprint arXiv:2406.15319*, 2024.

Jiazheng Kang, Mingming Ji, Zhe Zhao, and Ting Bai. Memory os of ai agent. *arXiv preprint arXiv:2506.06326*, 2025a.

Jikun Kang, Wenqi Wu, Filippos Christianos, Alex James Chan, Fraser David Greenlee, George Thomas, Marvin Purtorab, and Andrew Toulis. Lm2: Large memory models. *arXiv preprint arXiv:2502.06049*, 2025b.

Chris Latimer, Nicolo Boschi, Andrew Neeser, Chris Bartholomew, Gaurav Srivastava, Xuan Wang, and Naren Ramakrishnan. Hindsight is 20/20: Building agent memory that retains, recalls, and reflects. *arXiv preprint arXiv:2512.12818*, 2025.

Patrick Lewis, Ethan Perez, Aleksandra Piktus, Fabio Petroni, Vladimir Karpukhin, Naman Goyal, Heinrich Küttler, Mike Lewis, Wen-tau Yih, Tim Rocktäschel, et al. Retrieval-augmented generation for knowledge-intensive NLP tasks. *Advances in Neural Information Processing Systems*, 33:9459–9474, 2020.

Kai Li, Xuanqing Yu, Ziyi Ni, Yi Zeng, Yao Xu, Xin Zhang, Jitao Li, Xiaogang Sang, Xuelei Duan, Xiaogang Wang, et al. Timem: Temporal-hierarchical memory consolidation for long-horizon conversational agents. *arXiv preprint arXiv:2601.02845*, 2026.

Jiaqi Liu, Yaofeng Su, Peng Xia, Siwei Han, Zeyu Zheng, Cihang Xie, Mingyu Ding, and Huaxiu Yao. Simplemem: Efficient lifelong memory for llm agents. *arXiv preprint arXiv:2601.02553*, 2026.

Nelson F Liu, Kevin Lin, John Hewitt, Ashwin Paranjape, Michele Bevilacqua, Fabio Petroni, and Percy Liang. Lost in the middle: How language models use long contexts. *Transactions of the Association for Computational Linguistics*, 12:157–173, 2024.

Adyasha Maharana, Dong-Ho Lee, Sergey Tulyakov, Mohit Bansal, Francesco Barbieri, and Yuwei Fang. Evaluating very long-term conversational memory of llm agents. *arXiv preprint arXiv:2402.17753*, 2024.

Ali Modarressi, Ayyoob Imani, Mohsen Fayyaz, and Hinrich Schütze. RET-LLM: Towards a general read-write memory for large language models. *arXiv preprint arXiv:2305.14322*, 2023.

Jiayan Nan, Wenquan Ma, Wenlong Wu, and Yize Chen. Nemori: Self-organizing agent memory inspired by cognitive science. *arXiv preprint arXiv:2508.03341*, 2025.

Charles Packer, Vivian Fang, Shishir G Patil, Kevin Lin, Sarah Wooders, and Joseph E Gonzalez. Memgpt: Towards llms as operating systems. *arXiv preprint arXiv:2310.08560*, 2023.

Joon Sung Park, Joseph C O'Brien, Carrie Jun Cai, Meredith Ringel Morris, Percy Liang, and Michael S Bernstein. Generative agents: Interactive simulacra of human behavior. In *Proceedings of the 36th Annual ACM Symposium on User Interface Software and Technology*, pages 1–22, 2023.

Ofir Press, Noah A Smith, and Mike Lewis. Train short, test long: Attention with linear biases enables input length extrapolation. In *International Conference on Learning Representations*, 2022.

Preston Rasmussen, Pavlo Paliychuk, Travis Beauvais, Jack Ryan, and Daniel Chalef. Zep: A temporal knowledge graph architecture for agent memory. *arXiv preprint arXiv:2501.13956*, 2025.

Sahil Sen, Elias Lumer, Anmol Gulati, and Vamse Kumar Subbiah. Chronos: Temporal-aware conversational agents with structured event retrieval for long-term memory. *arXiv preprint arXiv:2603.16862*, 2026.

Zheng Wang, Shu Teo, Jieer Ouyang, Yongjun Xu, and Wei Shi. M-rag: Reinforcing large language model performance through retrieval-augmented generation with multiple partitions. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1966–1978, 2024.

Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Fei Xia, Ed Chi, Quoc V Le, and Denny Zhou. Chain-of-thought prompting elicits reasoning in large language models. *Advances in Neural Information Processing Systems*, 35:24824–24837, 2022.

Di Wu, Hongwei Wang, Wenhao Yu, Yuwei Zhang, Kai-Wei Chang, and Dong Yu. Longmemeval: Benchmarking chat assistants on long-term interactive memory. In *International Conference on Learning Representations*, 2025.

Menglin Xia, Xuchao Zhang, Shantanu Dixit, Paramaguru Harimurugan, Rujia Wang, Victor Ruhle, Robert Sim, Chetan Bansal, and Saravan Rajmohan. Memora: A harmonic memory representation balancing abstraction and specificity. *arXiv preprint arXiv:2602.03315*, 2026.

Wujiang Xu, Zujie Liang, Kai Mei, Hang Gao, Juntao Tan, and Yongfeng Zhang. A-MEM: Agentic memory for llm agents. In *Advances in Neural Information Processing Systems*, volume 38, 2025.

Lianmin Zheng, Wei-Lin Chiang, Ying Sheng, Siyuan Zhuang, Zhanghao Wu, Yonghao Zhuang, Zi Lin, Zhuohan Li, Dacheng Li, Eric Xing, et al. Judging llm-as-a-judge with mt-bench and chatbot arena. *Advances in Neural Information Processing Systems*, 36:46595–46623, 2023.

Wanjun Zhong, Lianghong Guo, Qiqi Gao, He Ye, and Yanlin Wang. Memorybank: Enhancing large language models with long-term memory. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 38, pages 19724–19731, 2024.

A Related Work

We organize related work along the progression from context-window extension to retrieval-augmented generation and finally to memory-augmented generation, then discuss the external-service paradigm that ByteRover departs from.

Context-Window Extension. A direct line of work extends the effective context length of Transformers by modifying attention or positional extrapolation. Longformer [Beltagy et al., 2020] introduces sparse attention patterns, while ALiBi [Press et al., 2022] enables length extrapolation by injecting distance-aware linear biases into attention scores. More recently, LM2 [Kang et al., 2025b] proposes a decoder-only architecture augmented with auxiliary memory. While these approaches improve long-range coverage, they do not address the continual, evolving, and write-back nature of agent memory.

Retrieval-Augmented Generation. RAG [Lewis et al., 2020] augments an LLM with external retrieval over a fixed corpus. LongRAG [Jiang et al., 2024] studies integration with long-context LLMs, while M-RAG [Wang et al., 2024] uses multiple partitions for fine-grained retrieval. However, standard RAG typically assumes a static knowledge base, whereas agentic settings require memory that is continuously updated [Gutiérrez et al., 2025].

Memory-Augmented Generation. MAG systems maintain a time-variant memory that evolves via a feedback loop [Zhong et al., 2024, Park et al., 2023]. Recent work spans multiple architectural categories [Jiang et al., 2026b]: graph-structured memory (MAGMA [Jiang et al., 2026a], Zep [Rasmussen et al., 2025]), hierarchical tiers (MemGPT [Packer et al., 2023], MemoryOS [Kang et al., 2025a], EverMemOS [Hu et al., 2026]), entity-centric stores (A-MEM [Xu et al., 2025], Mem0 [Chhikara et al., 2025]), and episodic/reflective designs (Nemori [Nan et al., 2025]). All of these operate as *external services* that agents call into. ByteRover departs from this paradigm by embedding memory operations directly in the agent's reasoning loop.

Benchmarks and Evaluation. LoCoMo [Maharana et al., 2024] evaluates long-term conversational memory across 35 sessions with temporal and causal reasoning questions. LongMemEval [Wu et al., 2025] stress-tests memory retention at context lengths exceeding 100K tokens. Recent analysis [Jiang et al., 2026b] reveals that many benchmarks risk context saturation—fitting within modern 128K+ windows—and that lexical metrics (F1, BLEU) systematically diverge from semantic correctness, motivating the use of LLM-as-a-Judge evaluation [Zheng et al., 2023].

B Hyperparameter Configuration

C Context Tree Entry Example

Figure 3 shows a complete knowledge entry as stored on disk in the *Context Tree*.

Module	Parameter	Value
Search Index	MiniSearch version	v7
	Fields	title (5×), content (1×), path (1.5×)
	Max retrieval results	32
	Max content length	8,000 chars
Search Config	Fuzzy ratio	0.2
	Prefix matching	enabled
	Score normalization	$s/(1+s)$
Direct Response	High confidence threshold	0.93
	Minimum score	0.85
	Minimum gap (top vs #2)	0.08
OOD Detection	Minimum relevance score	0.6
	Unmatched term threshold	0.85
Lifecycle	Importance decay (daily)	$0.995^{\Delta t}$
	Recency decay	$e^{-\Delta t/30}$
	Maturity thresholds	draft < 35, core ≥ 85
Curation	Max files per operation	5
	Max chars per file	40,000
Compression	Level 1 (LLM)	Normal summarization
	Level 2 (Aggressive)	0.6× token budget
Cache	Query cache TTL	0 (disabled)
	Fingerprint cache TTL	0 (disabled)
Inference	LLM Backbone (curate/query)	Gemini 3 Flash
	Temperature	0.0
Judge	Model	Gemini 3 Flash
	Max tokens	8,192
	Thinking budget	0 (disabled)
	Prompts	Hindsight (verbatim)
Justifier	Model	Gemini 3.1 Pro
	Max tokens	32,768
	Thinking budget	low
	Prompts	Hindsight (adapted for LongMemEval)

Table 7: Hyperparameter configuration for ByteRover and evaluation pipeline.

.brv/context-tree/architecture/module_boundaries/auth_billing_cycle.md

```
---
title: Auth-Billing Circular Dependency
tags: [architecture, circular-dependency, tech-debt]
keywords: [auth, billing, import-cycle, tree-shaking]
related:
  - architecture/module_boundaries/auth_service_deps.md
  - tech_debt/prioritization/q1_2026_assessment.md
importance: 82
maturity: validated
recency: 0.91
accessCount: 7
updateCount: 3
createdAt: 2026-02-03T11:20:00Z
updatedAt: 2026-02-15T09:45:00Z
---

## Relations
@architecture/module_boundaries/auth_service_deps.md
@architecture/module_boundaries/billing_integration.md
@tech_debt/prioritization/q1_2026_assessment.md

## Raw Concept
**Task:** Map circular dependency between auth, billing,
and user-management modules after v1.8 release.
**Changes:** PR #847 introduced auth -> billing import.
**Files:** src/auth/middleware.ts, src/billing/subscriptionCheck.ts
**Timestamp:** 2026-02-03T11:20:00Z
**Author:** architecture-agent

## Narrative
### Structure
The dependency cycle forms a triangle:
auth -> billing -> user-management -> auth.

### Rules
Circular deps with runtime imports are severity: high.
Type-only circular imports are severity: low.
```

Figure 3: A complete knowledge entry in the *Context Tree*, showing the YAML frontmatter with lifecycle metadata, explicit relation annotations, raw concept (provenance), and narrative (interpreted structure).