Memory Architecture

NEW IN 2026

Agent memory matured from "pick a vector database" into a benchmarked production discipline in 2026, with dedicated evaluation suites (LoCoMo (Long Conversation Memory)), MemBench, MemoryAgentBench, MemoryArena) and an ecosystem spanning roughly 20+ frameworks and vector stores across managed-cloud, self-hosted, and local-MCP hosting models.

Type	Storage	Layer	Purpose
Short-Term	Session	In-memory	Conversation state
Long-Term	Cross-session	Vector DB	Semantic recall
Graph Memory	Persistent	Graph DB	Relationships
Episodic	Persistent	DB	Prior tasks / outcomes
Procedural	Persistent	DB	Learned workflows
Policy Memory	Persistent	DB	Rules & restrictions
Audit Memory	Permanent	SQL	Logs & traceability

Three Patterns for Memory Control

- Pattern A — Context-resident: everything lives in the context window with compression. Simple, but caps out fast on long-running agents.

- Pattern B — Retrieval-augmented (workhorse pattern): working memory in-context, long-term records in a vector or structured store, injected each step. Recommended default, the engineering burden is manageable and the main challenge is retrieval quality.

- Pattern C — Tiered memory with learned control: multiple tiers (context, structured DB, vector store, cold archive) managed by a learned or prompted controller. Highest headroom, highest engineering cost — graduate to this only when data shows Pattern B is the bottleneck.

Best Practice

Memory entries should include:

- source
- timestamp
- confidence
- owner
- retention policy
- trust score

Known open problem across 2026 memory systems: selective forgetting. Most benchmarked systems handle retrieval and test-time learning reasonably well but still fail conspicuously at deciding what to evict. Budget explicit engineering time for eviction policy, not just ingestion.

Revision #1
Created 2026-08-24 12:02:57 UTC by Aisha M. Nur
Updated 2026-08-24 12:03:15 UTC by Aisha M. Nur