ANSWERING IS NOT REPAIRING: QUERYABLE WORLD MEMORY FOR CHANGING SOURCES

Anonymous authors
Paper under double-blind review

ABSTRACT

Agents repeatedly interpret heterogeneous observations, but later queries may arrive after either the source or the task has changed. We ask what should survive compression so that interpreted state remains reusable and its applicability can be reassessed. Queryable World Memory (QWM) has two conceptual components: budgeted selective compilation of observations into persistent records, and query-time reassessment of whether a record should be reused, checked, or replaced. Records separate *answer state* from *repair certificates* that identify re-observable support. Controlled SQL and synthetic longitudinal replays show that certificates can localize targeted rereading. In the longitudinal replay, QWM reaches 81/86 exact answers versus 69/86 for full rereading while using 79.2% fewer supplied observation units. A one-transition replay over natural PyPI and GitHub histories finds similar observed accuracy to full rereading (139/168 versus 140/168) with 10.1% fewer input tokens, but lower accuracy than answer-only memory (149/168). On the 22 certified cells, every evaluated arm is correct, while QWM uses 9,345 input tokens versus 64,554 for full rereading. The measured coverage–repair frontier quantifies one cost of reusable interpretation: support evidence saves rereading where retained but displaces answer content. The three main comparisons do not verify a learned writer, new-query generalization, or persisted multistep updates; an exploratory appendix reports a bounded synthetic execution of a fixed model writer without upgrading those claims.

1 INTRODUCTION

Tool-using agents revisit the same external world through database results, documents, reports, webpages, and APIs. When interpretation remains transient, later tasks repeat retrieval and parsing; retaining every raw observation merely shifts the burden to storage. We study memory as reusable interpretation between observations and downstream query, update, or application. Our central question is: *what should be compiled from observations so that an agent can reassess and reuse interpreted state when either the source or the task changes?*

QWM organizes this problem into two components. *Budgeted selective compilation* chooses which interpretations and supports to retain without access to future tasks. *Applicability reassessment* asks at query time whether a retrieved record fits the requested meaning, time range, and current source state. Its outcome may be reuse, targeted checking, broader reobservation, or abstention. Within a record, *answer state* carries reusable payload, while a *repair certificate* carries scope, locators, and dependencies that can direct a later check. The distinction is useful under both task change, which can make an old interpretation irrelevant, and source change, which can make it stale.

As a hypothetical extension beyond the measured evidence, consider a nonclinical longitudinal record in which a report first records a value as 4.2 and later corrects it to 4.8. A first query may ask for the latest value; a different later query may ask for the trend across visits. Selective compilation preserves the interpreted event and its support, while reassessment checks both the correction and whether the retained event is sufficient for the new trend query.

The question is narrower than whether structured memory is useful. Existing systems already extract and consolidate facts, organize linked notes, maintain temporal graphs, and learn memory operations (Rasmussen et al., 2025; Xu et al., 2025; Chhikara et al., 2025; Yu et al., 2026; Shu et al., 2026). Budgeted memory writing, provenance-linked escalation, and modular memory evaluation are

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

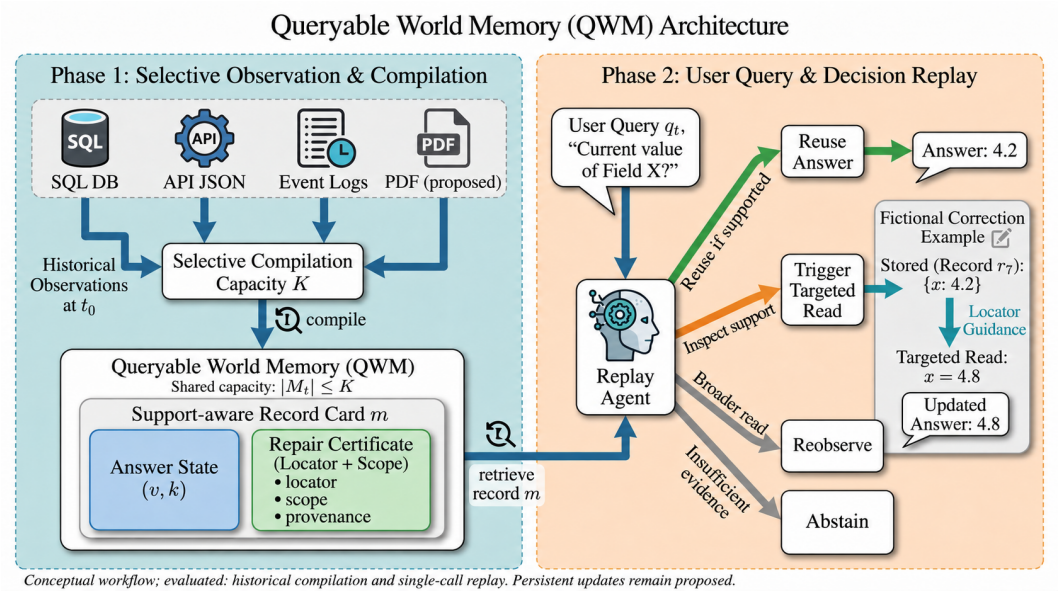


Figure 1: Conceptual QWM architecture. Answer state and support share capacity K. The correction is fictional; certificates guide reads but do not guarantee freshness or reveal hidden changes. Main experiments evaluate historical compilation and single-call replay; PDF inputs, persistent updates, and cross-task reassessment remain proposed.

also active topics (Bhargava & Sobral Barrento, 2026; Zhu et al., 2026; Zhou et al., 2026). Neither query-time probing nor structured caching is itself our contribution, and QWM has no advantage over an equally informative cache with the same policy. We instead isolate the compilation–reassessment contract and measure one consequence: how fixed capacity is divided between answer coverage and repair evidence.

We call this representation-policy contract *Queryable World Memory* (QWM). Figure 1 summarizes this contract. Typed adapters map observations to a common record while preserving source-specific locators and dependencies. Compilation and reassessment are backend agnostic and may be implemented by deterministic rules or a fixed model; training a writer is not required by the research question.

We contribute the typed QWM contract and evaluate controlled replay instantiations that measure the fixed-capacity answer–repair trade-off; we do not claim an evaluated online controller. Matched two-model replays cover controlled SQL changes, synthetic longitudinal records, and a natural PyPI–GitHub release transition. Certificates reduce observation on covered tasks, while population results depend on how capacity is divided between answer coverage and repair evidence.

2 Related Work and Novelty Boundary

Agent-memory systems already extract facts, consolidate experiences, organize temporal graphs, and learn memory operations. Zep integrates conversations and structured business data in a temporal knowledge graph (Rasmussen et al., 2025); A-MEM builds linked structured notes (Xu et al., 2025); Mem0 studies extracted conversational and graph memories (Chhikara et al., 2025); REMem performs iterative retrieval over episodic gists and facts (Shu et al., 2026); and AgeMem learns store, retrieve, update, summarize, and discard actions (Yu et al., 2026). LongMemEval evaluates extraction, multi-session and temporal reasoning, updates, and abstention over interaction histories (Wu et al., 2025). MEMAUDIT and a recent agent-native memory study further establish budgeted writing and modular evaluation as active problems (Bhargava & Sobral Barrento, 2026; Zhou et al., 2026). QWM studies how retained interpretation can be reassessed against external sources. The executed experiments isolate one part of that problem: the capacity tradeoff between state that answers now and evidence that identifies what to inspect after change.

Freshness and verification are also established concerns. FreshLLMs retrieves current search evidence at inference time (Vu et al., 2024); Grounding Agent Memory checks candidate memories with least-privilege world tools before committing them (Suresh et al., 2026); and TierMem links lossy summaries to immutable evidence and escalates when summary evidence is insufficient (Zhu et al., 2026). TemporalWiki builds evolving corpora from consecutive Wikipedia and Wikidata snapshots (Jang et al., 2022), while PAT-Questions refreshes present-anchored answers through knowledge-graph queries (Meem et al., 2024). WikiRecentChanges shows that recognizing temporal conflict does not reliably change a model's final prediction (Wallat et al., 2026). These works motivate natural-change evaluation and establish query-time evidence retrieval, provenance links, temporal validity, and selective verification as existing components; we do not claim any of these components alone as a contribution.

Under one capacity, bytes that preserve an answer compete with bytes that identify the support to reread after an external source changes. A conventional structured cache with the same answer state, repair certificate, and policy is behaviorally equivalent to QWM, so backend branding cannot explain an effect. The database replay isolates controlled repair; the longitudinal experiment stresses corrections, late arrivals, conflicts, and rule changes; and the natural-source stream uses immutable public release histories. Synthea provides a synthetic-health-data generation precedent (Walonoski et al., 2018); MedMemoryBench provides a medical agent-memory evaluation precedent (Wang et al., 2026); and FHIR-AgentBench provides an interoperable EHR question-answering precedent (Lee et al., 2026). Together, they motivate the longitudinal adapter. These experiments measure an answer–repair frontier within the broader compilation-and-reassessment problem; they do not establish a new storage engine or generalization to unseen queries.

3 Queryable World Memory

QWM separates two decisions that are often conflated: what interpretation to retain when observations arrive, and whether that interpretation applies when a query arrives. The first is *budgeted selective compilation*; the second is *applicability reassessment*. The measured coverage–repair frontier is a cost of implementing these decisions under finite capacity, not the definition of the broader problem.

3.1 Online information boundary

Let X_t be hidden world state and q_t the current task. Sources expose observations with content, locator, metadata, observation time, and resource cost; observation time is not valid time unless the source says so. At decision step j, the policy may use only

$$I_{t,j} = \{q_t, \rho(M_t, q_t), o_{t,1:j-1}, B_{t,j}\}, \qquad |M_t| \le K, \tag{1}$$

where ρ retrieves from bounded memory, $o_{t,1:j-1}$ are newly purchased observations, and $B_{t,j}$ is remaining budget. Prior raw observations require retention or charged reobservation; hidden mutations, evaluator answers, and future tasks are unavailable. Identical accessible information can mask different worlds, so freshness requires new observation or an assumption about change signals.

3.2 Component 1: budgeted selective compilation

A source adapter maps observed content into

$$m = (v, k, \sigma, S, p, t_s, t_o, \nu, D, r, h). \tag{2}$$

Here (v, k) are payload and retrieval keys; σ is applicability scope; (S, p) identify support and provenance; (t_s, t_o, ν) capture source time, observation time, and exposed version; and (D, r, h) hold dependencies, validation state, and update history. Evaluated adapters use SQL read sets; longitudinal event, field, time, and rule dependencies; and API-field or release-document supports. Supported-at-observation does not imply current validity, and conflicting records retain their scopes.

The *answer state* principally contains (v, k). A *repair certificate* draws from the remaining fields to direct a later check; no minimality is claimed. The distinction is logical, so an equally informative structured cache with the same policy is equivalent.

Selective compilation chooses payloads and certificates under capacity. Let binary x_i retain payload unit i and binary c_i its certificate, with charged byte costs p_i and d_i. For historical support bundle S_b,

binary a_b and r_b indicate complete payload and certificate coverage, and g_b estimates observations avoided by a usable certificate:

$$\begin{aligned} \max_{x,c,a,r} \quad & \sum_b a_b + \beta \sum_b g_b r_b \\ \text{s.t.} \quad & \sum_i (p_i x_i + d_i c_i) \le K, \quad c_i \le x_i, \\ & a_b \le x_i, \quad r_b \le c_i \qquad\qquad \forall b,\ i \in S_b. \end{aligned} \tag{3}$$

Stage 1D uses multi-unit bundles with $g_b = (|S_b|-1)_+$ under its declared intervention model. Stage 3 instead allocates singleton records with $g_i = \max(3 - |D_i|, 0)$, where three is a nominal reread proxy and some projects expose only two sources. Allocation is one implementation of selective compilation, not a separate headline contribution.

3.3 Component 2: applicability reassessment

In the proposed contract, retrieval proposes records and applicability reassessment checks whether their keys and scope fit the requested meaning, entity, and time range, and whether their supports remain adequate under observed source state. A task change can make a fresh record irrelevant or insufficient; a source change can make an otherwise relevant record stale. The policy may then choose Reuse, targeted Probe, validated Repair, broad Reobserve, or Abstain. QWM connects these familiar actions to compiled records; the tested contribution is the measured evidence below, not a novelty claim about this wiring.

Let C_K compile observations into memory of capacity K. For query distribution P_q, define answerability as

$$A_K(C) = 1 - \mathbb{E}_{q \sim P_q}[L_{\text{task}}(q, C_K(O))]. \tag{4}$$

After relevant source change δ, let $R_K(C, \delta; \epsilon)$ be the additional observation and computation required to restore expected task loss below ϵ. The pair (A_K, R_K) measures one tension in reusable interpretation: support-aware state may lower rereading cost while displacing payload. It does not measure whether the system generalizes to unseen tasks.

3.4 Current implementation boundary

The three main comparisons instantiate both components narrowly. They deterministically construct historical records from exact old-snapshot answers, preventing future leakage but idealizing writing. The harness supplies arm-specific observation views before a single model call; the model produces an answer or SQL response, rather than autonomously selecting a sequence of source reads. In the natural track, retained certificates determine targeted views and other records receive full-project views. They do not verify general scope checking, a learned writer, unseen-query generalization, or durable state updates across episodes. Appendices A.1 and B separately execute fixed writers over three- and sixteen-packet synthetic streams. These studies measure compilation, persistence and maintenance under explicit arriving corrections, not hidden-drift detection or query-time applicability reassessment. Known query types, small contexts and DEV-guided protocols preclude general writing, new-query transfer or repair-certificate claims.

4 Evaluation Protocol

We evaluate a fixed-capacity trade-off between answer coverage and repair evidence in three tracks. SQL repair is conditioned on 12 certificate-retained targets, synthetic longitudinal repair on 12 certificate-retained targets, and natural software-release utility is measured on all 84 project-holdout pairs regardless of retention. Thus only the natural track estimates population utility under the frozen allocation. All scheduled cells use one model call with no retries. The experiments test recovery from deterministically constructed historical records; they do not test learned extraction or persistent updates across multiple transitions.

4.1 QUESTIONS AND CONTROLLED SQL REPLAY

We ask: (1) how answer-only and certificate-bearing records trade coverage against repair; (2) whether frozen model policies can use targeted dependencies instead of broad rereading; and (3) whether behavior changes across policies and adapters. Lifecycle amortization after learned compilation, maintenance, serialization, restart, and later reuse remains open.

The SQL workload is derived from BIRD (Li et al., 2023). The replay uses D0 (no change), D1 (query-relevant column rename), and D2 (irrelevant rename). The agent does not receive the condition, rename map, expected rows, or future evaluator reference SQL; the historical-procedure arm intentionally retains old SQL. All persistent views are compiled from the same 110-history workload. For database d, $K_d = \lfloor 0.5 B_d \rfloor$, where B_d is the serialized size of all answer-only histories; 30% and 70% are offline sensitivities. Five paid arms are full-schema rereading, bounded text summary, historical-SQL reparse, answer-only state, and answer state plus certificate. A structured cache with the same state, certificate, and policy is an offline exact-match control.

Coverage uses all 110 histories. Repair uses 12 targets selected before model outcomes from the certificate-retained pool, balanced 4/4/4 by benchmark difficulty across ten databases; this conditioning precludes a population-effectiveness interpretation. Crossing two frozen model policies, five arms, and three conditions yields 360 calls. Predicted SQL is executed against the intervened SQLite database and compared by typed ordered result. These are previously observed procedures from a rename-eligible pool, not unseen NL-to-SQL prediction or an official BIRD score.

4.2 LONGITUDINAL AND NATURAL-CHANGE TRACKS

The synthetic longitudinal track contains structured rows, raw reports, and event notes with correction, late-arrival, conflict, irrelevant-update, and rule-version changes. No real patient data or treatment task is included. At 50% split-workload capacity, answer-only and certificate views retain 51/96 and 12/96 records. Twelve certificate-retained targets produce 43 target–condition cells; 11 are also answer-only-retained and one is a declared boundary case. Four arms and two policies give 344 calls. Earlier prompt-development runs are excluded; the final replay and exact typed scorer are process-separated.

The natural track uses earlier and later stable releases from 24 public projects. Versioned PyPI JSON provides package metadata and hashes; GitHub releases provide tags, timestamps, and document text. Seven predefined tasks cover version, Python requirement, dependencies, source-distribution hash, release-note sections, cross-source version consistency, and publication order. Projects are disjoint between DEV and PROJECT-HOLDOUT; the proposed surface and task holdouts were not implemented.

Historical records are extracted deterministically from the old snapshot. Writing and reading share the seven task types and exact `family_id:task` lookup. The text-summary baseline is a JSON rendering of the old answer, not an LLM summarizer or vector-memory system. These choices isolate recovery but do not test a learned compiler or unseen-query retrieval.

Stage 3 constrains the *logical sum* of canonical selected components, not the serialized enclosing view. Each answer and certificate component is charged separately; catalog sharing and container overhead affect only the diagnostic serialized size. On holdout, $K = 7016$ bytes: answer-only uses 6942 logical bytes (7017 serialized), and QWM 6990 (6130 serialized). Answer-only maximizes retained records. QWM solves a singleton specialization of Equation 3: an answer earns one unit and a certificate earns $\beta \max(3 - |D_i|, 0)$ with $\beta = 1.5$. This declared reward is not measured savings. Certified records reread declared supports; others use full-project rereading. Four paid arms and two policies yield 672 calls; the exact matched cache aliases QWM outputs.

4.3 OUTCOMES AND INFERENCE

We report exact task correctness, retention coverage, supplied observation units/bytes, model tokens, storage under each track's declared convention, and provider cost. Observation units count supplied event rows in the longitudinal track and source objects in the natural track, not physical I/O or API requests; SQL reports exposed schema rows. These units are not comparable across adapters. The three main tracks do not measure extraction, persistent maintenance, restart/reload or end-to-end

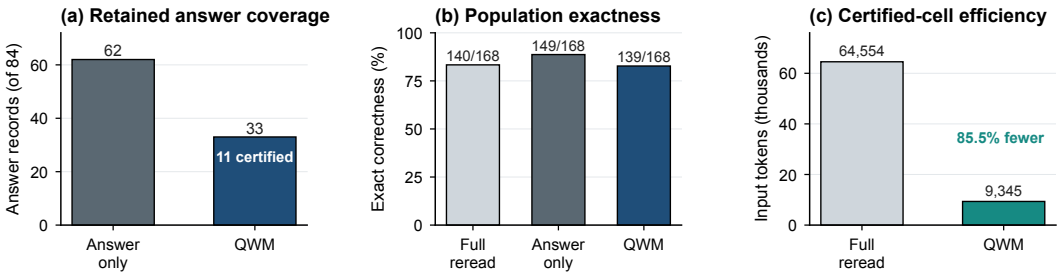


Figure 2: Natural software-release results. Under the same 50% capacity, QWM retains fewer answers than answer-only memory (a) and is less accurate over the population (b), yet uses 85.5% fewer input tokens than full rereading on 22 certified model cells (c). Panel (c) is conditional on allocation.

latency. Supplemental DEV studies separately measure model-written maintenance, not hidden-change discovery or query-time revalidation. Appendix A records frozen settings and capacity accounting.

Families or projects, not cells or model policies, are independent units. Cell-level McNemar tests are exploratory. For the natural track, a fixed-seed bootstrap resamples the 12 holdout projects as clusters for paired accuracy and resource intervals; with 12 projects, these intervals describe project heterogeneity and do not establish equivalence. Prompts, tool views, exclusions, model routes, and budgets were frozen before each formal run.

5 Results

Figure 2 summarizes the natural holdout: certificates save rereading where allocated, but their capacity cost reduces answer coverage and does not improve population accuracy.

5.1 Offline mechanism and allocation checks

Synthetic DEV checks show that the mechanism is implementable, not that it is superior. In three hand-authored SQL families, no-memory, unvalidated-cache, and support-aware policies are correct in 6/6, 3/6, and 6/6 episodes. Larger fixed-byte suites reproduce the coverage–repair trade-off, but a versioned/dependency cache with identical information and policy is exactly identical to QWM.

A future-isolated check uses 923 historical and 611 future BIRD DEV queries (Li et al., 2023) across 11 databases. At 20% capacity, historical support-bundle selection raises macro complete-support answerability from 0.290 to 0.409 over item LFU (positive in 8/11 databases), but the advantage reverses at larger capacities. Complete certificates reduce declared repair observations while sacrificing answerability. All 176 database–capacity–β MILPs solve optimally; a greedy heuristic misses the 0.90 objective-ratio gate in 5/16 aggregate cells and is rejected.

A stable scan accepts 110 procedure families from 123 attempts. Eligibility requires stale-SQL failure after rename and deterministic repair success, so 110/110 failures and restorations are selected-population mechanism checks, not general recovery rates. Certificates observe 26.373 schema rows on average versus 72.545 for full validation; the matched cache is identical in all 330 cells.

5.2 Controlled model replays

At 50% per-database capacity, answer-only memory retains 51/110 SQL histories and certificate memory 33/110. Repair evaluation conditions on 12 certificate-retained procedures. The two-model, five-arm, three-condition replay comprises 360 calls. Under no or irrelevant change, answer-only, certificate, and SQL-reparse are each 24/24 correct. Under relevant rename, certificate is 16/24, versus answer-only 2/24, reparse 5/24, full rereading 4/24, and text summary 0/24. Certificate validation observes 24.2 schema rows on average, versus 64.3 for answer-only recovery and 70.6 for reparse. `deepseek-flash` repairs 12/12 certificate cases; `qwen3.7-plus-2026-05-26` repairs 4/12, abstains in six, and emits two unexecutable answers. Certificates improve conditional repair but

Table 1: SQL replay exact correctness, pooled across two models; 24 episodes per arm and condition.

Arm	D0	D1	D2	D1 rows
Full reread	12/24	4/24	12/24	64.3
Text summary	11/24	0/24	11/24	64.3
SQL reparse	24/24	5/24	24/24	70.6
Answer only	24/24	2/24	24/24	64.3
Answer + certificate	24/24	**16/24**	24/24	**24.2**

Table 2: Synthetic longitudinal replay pooled across two models; Obs. is mean supplied event rows per episode.

Arm	All exact	Relevant	Obs.	Input tokens
Full reread	69/86	24/38	9.72	1873
Text summary	81/86	33/38	9.72	1986
Answer only	80/86	32/38	7.42	1654
Answer + certificate	**81/86**	**34/38**	**2.02**	**1163**

neither remove policy uncertainty nor provide free coverage. Table 1 reports the pooled arm-level results.

The longitudinal replay completes all 344 calls. Certificates are 81/86, versus full rereading 69/86 ($p = 0.0075$), text summary 81/86, and answer-only 80/86 ($p = 1.0$). On 38 relevant changes, certificate/full is 34/38 versus 24/38 ($p = 0.0129$). Certificate views reduce mean supplied observation units by 79.2% and input tokens by 37.9%. The full contrast reverses by policy: certificate/full is 39/43 versus 42/43 for `deepseek-flash`, but 42/43 versus 27/43 for `qwen3.7-plus-2026-05-26`. This supports targeted rereading and a policy interaction, not universal superiority or clinical validity. Table 2 reports the pooled results and resource means.

5.3 Natural software-release changes

All 672 calls finish; four malformed outputs remain incorrect. Over 168 model cells, QWM has similar observed accuracy to full rereading (139 versus 140) while reducing serialized observation-view bytes by 10.9% and input tokens by 10.1%. The project-cluster accuracy difference is -0.6 points (95% interval $[-3.6, 2.4]$); crossing zero does not establish equivalence or non-inferiority. The observation-view byte reduction misses the historical protocol's 30% target. Answer-only is 149/168; QWM's difference is -6.0 points ($[-8.3, -3.6]$). Stable-field correctness is 70/94 for QWM and 79/94 for answer-only; changed-field correctness is 69/74 and 70/74. Table 3 reports the pooled results and resource means.

Table 3: Natural project-holdout replay pooled across two models; resources are means per episode; Obs. counts supplied source objects, not network requests.

Arm	All exact	Changed	Obs.	Input tokens
Full reread	140/168	68/74	2.83	3159
Text summary	135/168	66/74	2.83	3185
Answer only	**149/168**	**70/74**	2.83	3186
QWM budgeted	139/168	69/74	**2.60**	**2839**

Retention stratification is consistent with coverage pressure but is not causal identification. In 22 certified common-retained cells, QWM and answer-only are both 22/22, although prompts differ. In 44 uncertified common-retained cells both are 39/44, and in 44 cells retained by neither both are 40/44; all paired prompt messages are identical within those two strata. The entire ten-answer advantage of answer-only occurs in 58 coverage-discordant cells where answer-only retains a record and QWM does not (48/58 versus 38/58). A matched-retention intervention is needed to separate allocation from prompt and policy effects.

On the 22 certified cells, QWM uses 22 supplied observation units and 9,345 input tokens versus 62 and 64,554 for full rereading, an 85.5% input reduction. This establishes efficient targeted observation on a selected subset, not absence of targeted failures globally. Under the frozen logical-cost ceiling, QWM retains 33 answer records, 11 of which include certificates, versus 62 answer-only records. Local repair efficiency therefore does not imply population-level memory utility.

5.4 Exploratory persisted-writer execution

Appendices A.1 and B study persisted writers over three- and sixteen-packet streams. In the latter, provenance-retained exact states are 32/32 for both DeepSeek updates and 17/32 versus 32/32 for Qwen full/local updates; local output-token usage falls by about 85%. Correct latest values can hide historical identity errors. These are two dependent synthetic trajectories with known queries and no 50% capacity intervention, not lifecycle evidence.

Appendix C further matches source prefixes, byte ceilings and a shared reader for structured memory versus free-text summaries. Its prompt-revised DEV cells show no universal winner: capacity compliance and reader errors both affect outcomes. The two trajectories and repeated checkpoints remain dependent exploratory evidence.

6 Limitations and Conclusion

QWM frames world memory as two linked problems: selectively compile observations into reusable interpretation, then reassess applicability when either the source or the task changes. Separating answer state from repair certificates makes one cost measurable: controlled changes show cheaper targeted rereading where certificates are retained, while natural release histories expose the answer coverage they displace. This coverage–repair frontier is evidence about finite-capacity design, not the whole research question, and QWM does not outperform an equally informative cache with the same policy.

The evidence remains narrow. Partial observability prevents freshness guarantees; the controlled settings do not estimate natural change frequencies, and the natural track covers one transition in one software ecosystem. The three main comparisons verify neither a learned writer, genuinely new query types, nor persisted multistep repair, and they establish no clinical effectiveness or lifecycle break-even. Exploratory fixed-writer studies cover three-packet streams and two sixteen-packet trajectories. Local upserts reduce generated state, while full rewriting can corrupt event identity despite correct latest-value answers. Small synthetic samples, known query types and DEV-guided protocols preclude confirmatory or model-general claims. These studies do not isolate a repair-certificate benefit or execute the general query-time policy cycle. Persistent memory further requires source permissions, deletion, retraction, and untrusted-input handling. The next scientific step is to test the compilation–reassessment contract on held-out tasks and natural update sequences while charging maintenance, latency, failures, and revalidation.

AI use statement

Generative AI tools were used to propose and refine hypotheses, critique experimental designs, implement and refactor software, assist literature discovery, edit manuscript text, and revise figure layout and labels using image-generation tools. They also assisted the implementation of the deterministic generator for the synthetic longitudinal dataset; the requested model identifier `gpt-5.6-luna` via the Doro platform produced 15 DEV-only wording variants after the underlying facts and programmatic oracles were frozen. The requested identifiers `gpt-5.5`, `gpt-5.6-sol`, and `gpt-5.6-terra`, also via Doro, were evaluated as writers in the exploratory appendix, not as oracles; successful responses returned the same identifiers. Doro is the platform route, not a model name, and its proxy labels do not independently verify upstream checkpoints. Generative AI did not act as the oracle, choose natural release transitions based on outcomes, score model responses, or alter deterministic numerical results. Outputs from `deepseek-flash`, `qwen3.7-plus-2026-05-26`, and the three Doro-routed identifiers are evaluation objects rather than writing assistance. AI-assisted code was checked by executable tests and exact scorers, citations were checked against primary metadata, and claims were traced to hashed result artifacts. The authors take responsibility for the final text, claims, code, and artifacts.

Ethics statement

The longitudinal evaluation uses pseudonymous records generated by deterministic rules and contains no real patient data, human subjects, diagnoses, or treatment recommendations. The natural-change evaluation uses public package metadata and release documents. Persistent world memory can nevertheless retain sensitive, stale, or access-controlled information; deployment would require source-level authorization, deletion and retention policies, provenance-aware correction, and auditing beyond the mechanisms evaluated here. API credentials and private provider traces are excluded from released artifacts.

Reproducibility statement

The anonymous offline supplement reaggregates saved scorer decisions for the three main comparisons and audits saved outputs for Appendix C, with hashed manifests. It does not independently reground main-comparison answers or reproduce model inference; external SQL databases and natural-source payloads are excluded pending redistribution clearance. Main-comparison gold answers are excluded from model inputs; the appendix's explicitly controlled-prior diagnostic supplies previous-step oracle state only. Settings and budgets are frozen per run. The database run uses 342,764 provider-reported tokens (USD 0.025997; CNY 0.390510), the longitudinal run 604,290 (USD 0.100515; CNY 0.719894), and the natural-release run 2,104,515 (USD 0.303852; CNY 2.344270).

References

Nishant Bhargava and Rodrigo Sobral Barrento. MEMAUDIT: An exact package-oracle evaluation protocol for budgeted long-term LLM memory writing. *arXiv preprint arXiv:2605.02199*, 2026. URL https://arxiv.org/abs/2605.02199.

Prateek Chhikara, Dev Khant, Saket Aryan, Taranjeet Singh, and Deshraj Yadav. Mem0: Building production-ready AI agents with scalable long-term memory. In *ECAI 2025*, pp. 2993–3000, 2025. doi: 10.3233/FAIA251160. URL https://journals.sagepub.com/doi/10.3233/FAIA251160.

Joel Jang, Seonghyeon Ye, Changho Lee, Sohee Yang, Joongbo Shin, Janghoon Han, Gyeonghun Kim, and Minjoon Seo. TemporalWiki: A lifelong benchmark for training and evaluating ever-evolving language models. In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing*, pp. 6237–6250, 2022. doi: 10.18653/v1/2022.emnlp-main.418. URL https://aclanthology.org/2022.emnlp-main.418/.

Gyubok Lee, Elea Bach, Eric Yang, Tom Pollard, Alistair Johnson, Edward Choi, Yugang Jia, and Jong Ha Lee. FHIR-AgentBench: Benchmarking LLM agents for realistic interoperable EHR question answering. In *Proceedings of the Fifth Machine Learning for Health Symposium*, volume 297 of *Proceedings of Machine Learning Research*, pp. 1297–1315. PMLR, 2026. URL https://proceedings.mlr.press/v297/lee26a.html.

Jinyang Li, Binyuan Hui, Ge Qu, Jiaxi Yang, Binhua Li, Bowen Li, Bailin Wang, Bowen Qin, Ruiying Geng, Nan Huo, Xuanhe Zhou, Chenhao Ma, Guoliang Li, Kevin Chang, Fei Huang, Reynold Cheng, and Yongbin Li. Can LLM already serve as a database interface? a BIg bench for large-scale database grounded text-to-SQLs. In *Advances in Neural Information Processing Systems*, volume 36, 2023. URL https://proceedings.neurips.cc/paper_files/paper/2023/hash/83fc8fab1710363050bbd1d4b8cc0021-Abstract-Datasets_and_Benchmarks.html.

Jannat Meem, Muhammad Rashid, Yue Dong, and Vagelis Hristidis. PAT-questions: A self-updating benchmark for present-anchored temporal question-answering. In *Findings of the Association for Computational Linguistics: ACL 2024*, pp. 13129–13148, 2024. doi: 10.18653/v1/2024.findings-acl.777.

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. URL https://arxiv.org/abs/2501.13956.

Yiheng Shu, Saisri Padmaja Jonnalagedda, Xiang Gao, Bernal Jimenez Gutierrez, Weijian Qi, Kamalika Das, Huan Sun, and Yu Su. REMem: Reasoning with episodic memory in language agent. In *The Fourteenth International Conference on Learning Representations*, 2026. URL https://openreview.net/forum?id=fugnQxbvMm.

Susheel Suresh, Hazel Mak, Sahil Bhatnagar, Chhaya Methani, and Alejandro Gutierrez Munoz. Grounding agent memory: Environment-probing curation for enterprise agents. *arXiv preprint arXiv:2609.11060*, 2026. URL https://arxiv.org/abs/2609.11060.

Tu Vu, Mohit Iyyer, Xuezhi Wang, Noah Constant, Jerry Wei, Jason Wei, Chris Tar, Yun-Hsuan Sung, Denny Zhou, Quoc V. Le, and Thang Luong. FreshLLMs: Refreshing large language models with search engine augmentation. In *Findings of the Association for Computational Linguistics: ACL 2024*, pp. 13697–13720, 2024. doi: 10.18653/v1/2024.findings-acl.813. URL https://aclanthology.org/2024.findings-acl.813/.

Jonas Wallat, Wolfgang Nejdl, and Sandipan Sikdar. When facts change: Temporal knowledge conflict resolution in LLMs. In *Findings of the Association for Computational Linguistics: ACL 2026*, pp. 2154–2184, 2026. doi: 10.18653/v1/2026.findings-acl.103.

Jason Walonoski, Mark Kramer, Joseph Nichols, Andre Quina, Chris Moesel, Dylan Hall, Carlton Duffett, Kudakwashe Dube, Thomas Gallagher, and Scott McLachlan. Synthea: An approach, method, and software mechanism for generating synthetic patients and the synthetic electronic health care record. *Journal of the American Medical Informatics Association*, 25(3):230–238, 2018. doi: 10.1093/jamia/ocx079.

Yihao Wang, Haoran Xu, Renjie Gu, Yixuan Ye, Xinyi Chen, Xinyu Mu, Yuan Gao, Chunxiao Guo, Peng Wei, Jinjie Gu, Huan Li, Ke Chen, and Lidan Shou. MedMemoryBench: Benchmarking agent memory in personalized healthcare. *arXiv preprint arXiv:2605.11814*, 2026. URL https://arxiv.org/abs/2605.11814.

Di Wu, Hongwei Wang, Wenhao Yu, Yuwei Zhang, Kai-Wei Chang, and Dong Yu. LongMemEval: Benchmarking chat assistants on long-term interactive memory. In *The Thirteenth International Conference on Learning Representations*, 2025. URL https://openreview.net/forum?id=pZiyCaVuti.

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. doi: 10.52202/085713-0593.

Yi Yu, Liuyi Yao, Yuexiang Xie, Qingquan Tan, Jiaqi Feng, Yaliang Li, and Libing Wu. Agentic memory: Learning unified long-term and short-term memory management for large language model agents. In *Proceedings of the 64th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 21457–21483, 2026. doi: 10.18653/v1/2026.acl-long.981. URL https://aclanthology.org/2026.acl-long.981/.

Wei Zhou, Xuanhe Zhou, Shaokun Han, Hongming Xu, Guoliang Li, Zhiyu Li, Feiyu Xiong, and Fan Wu. Are we ready for an agent-native memory system? *arXiv preprint arXiv:2606.24775*, 2026. URL https://arxiv.org/abs/2606.24775.

Qiming Zhu, Shunian Chen, Rui Yu, Zhehao Wu, and Benyou Wang. From lossy to verified: A provenance-aware tiered memory for agents. *arXiv preprint arXiv:2602.17913*, 2026. URL https://arxiv.org/abs/2602.17913.

A Reproducibility Details

Frozen model execution. Table 4 reports the executed request settings rather than current provider defaults. All three tracks used the direct-provider model identifiers `deepseek-flash` and `qwen3.7-plus-2026-05-26`. The former was recorded as the DeepSeek-V4.1-Flash service alias observed during execution, not an immutable weight revision; the latter is a dated Qwen snapshot. Thinking was disabled and temperature was zero for both models. The runner enforced zero client

Table 4: Frozen request configuration for the three executed model replays. Input/output are token ceilings per request.

Track	Calls	Input	Output	Timeout (s)	Retries
SQL	360	4096	512	60	0
Longitudinal	344	4096	1024	60	0
Natural release	672	16384	512	90	0

retries in every track, so malformed measured outputs remained in the denominator and non-contract transport failures failed closed.

The natural-track project-cluster bootstrap used seed `20270921` and 10,000 resamples of the 12 holdout projects. Cell-level model outputs were not treated as independent samples.

Natural transition rule. For each project, eligible releases had a valid version, were neither prereleases nor development releases, contained files, and had a timestamp from a non-yanked file. Candidates were ordered by upload timestamp and parsed version. The later snapshot was the latest eligible release. The earlier snapshot was the most recent eligible release whose major–minor series differed from the latest; if none existed, the immediately preceding eligible release was used. Selection used no model outcome.

Stage 3 capacity accounting. Let $b(a_i)$ be the canonical UTF-8 byte length of answer record i, and $b(c_i)$ the canonical byte length of its certificate component. The frozen capacity is $K = \lfloor 0.5 \sum_i b(a_i) \rfloor$. Answer-only charges $\sum_{i \in A} b(a_i)$; QWM charges $\sum_{i \in A} b(a_i) + \sum_{i \in C} b(c_i)$ with $C \subseteq A$. The serialized enclosing view is diagnostic and may differ because it adds container overhead while deduplicating dependencies into a shared catalog. Table 5 reports the exact values.

Table 5: Frozen Stage 3 bytes. Each arm cell is logical used / serialized view; full is the logical size of all answer records.

Split	Full	K	Answer-only	QWM
DEV	12093	6046	6000 / 6068	5984 / 5480
PROJECT-HOLDOUT	14032	7016	6942 / 7017	6990 / 6130

Retention diagnostics were computed from the frozen allocation and paired replay artifacts. The submitted supplement supports offline reaggregation of saved main-track scorer decisions; it does not include the complete retention-audit inputs or independently reproduce that stratification.

Logging limitation. The three main comparisons retain prompt hashes, parsed outputs, status, usage, and ledgers, but not verbatim response text. The exploratory writer runner additionally retains raw response text and per-call metadata.

A.1 Exploratory text-bound persisted writer

Four fictional families use four text styles, three packets, explicit corrections, and two known queries. Reports expose only `source_id`, `observed_time`, and `raw_text`. Persistence sees the current packet plus model state; rereading sees all reports and empty prior state. Gold and canonical metadata stay outside prompts and support only grouping and validation. Both emit full state to a fixed reader. Validation makes no semantic repair and normalizes only top-level `type: json_object`; process separation is not OS isolation.

In Table 6, DS denotes `deepseek-flash` and Qwen denotes `qwen3.7-plus-2026-05-26`.

Relative to rereading, persistence uses 10.27% fewer input tokens for DS and 7.39% fewer for Qwen. Serialized-byte diagnostics are retained in the artifacts and are not network bytes. Qwen's rejected output used a non-ISO time; no repair or retry was applied. Prior DEV smoke tests guided this exploratory protocol.

Table 6: Exploratory text-bound writer results. Exact state compares all event fields with the oracle; input is provider-reported tokens.

Writer	Memory mode	Exact state	Query correct	Input tokens
DS	Persistent	12/12	24/24	9,044
DS	Full reread	12/12	24/24	10,079
Qwen	Persistent	11/12	22/24	9,781
Qwen	Full reread	12/12	24/24	10,561

In the legacy 45-second `gpt-5.6-terra`-via-Doro stratum, persistence has 6/12 exact states but 24/24 known-query answers: three untouched dates shift and persist while values and identifiers remain intact. Rereading times out once, then blocks 11 calls. The reservation is not measured usage; no paired claim follows.

The 120-second extension in Table 7 uses Doro as a platform; exact IDs are proxy labels, not verified upstream checkpoints.

Table 7: Exploratory 120-second extension on the same fixture and prompt V3; P/R denote persistent/full reread.

Model ID	Platform	Mode	Exact state	Query correct	Input tokens
`gpt-5.5`	Doro	P	12/12	24/24	8,491
`gpt-5.5`	Doro	R	12/12	24/24	9,308
`gpt-5.6-sol`	Doro	P	8/12	24/24	8,625
`gpt-5.6-sol`	Doro	R†	9/12	18/24	6,187
`gpt-5.6-terra`	Doro	P	10/12	24/24	8,491
`gpt-5.6-terra`	Doro	R†	10/12	20/24	7,281

All rows use the Responses API, provider-default decoding, zero retries, and 12-state/24-query denominators. Rereading yields 9 responses/10 attempts and three failed cells for `gpt-5.6-sol`, versus 10/11 and two for `gpt-5.6-terra`. Daggers mark connection errors and fail-closed blocking; every accepted reread state is exact, so incomplete token and cost totals are not compared. `gpt-5.5` persistence uses 8.78% fewer input tokens. Observation-time errors recur at different counts, not an estimated frequency. The separate 45-second stratum's byte-identical first prompt reports 793 versus 497 input tokens; this observed token-count difference does not establish its cause.

B LONG-SEQUENCE MEMORY MAINTENANCE

Two fictional families each contain three entities and 16 packets: 40 original reports (10 late arrivals) and 20 corrections across four text styles. Each final state has 20 events; there is no 50% retention cap. Four queries inspect latest value, event value, source and observation time. The maintenance/reread comparisons exclude query instances and oracle state from writer inputs.

Full-state rewriting (Full) and mechanically merged upserts (Local) receive the current packet and their own committed memory. History instead receives all reports observed so far and empty prior memory. Table 8 retains provenance; DS denotes `deepseek-flash`, Qwen `qwen3.7-plus-2026-05-26`. Each row has 32 dependent state checkpoints and 128 queries.

Against Full, Local reduces output tokens by 85.1%/85.2% for DS/Qwen, with nearly unchanged input (slightly higher for Qwen). DS ties in accuracy. Qwen's 15 inaccurate Full checkpoints lie on one trajectory from packet 2 onward, not 15 independent errors. Rewriting substitutes correction-report IDs for stable event IDs and omits one initially correct untouched event. All latest-value queries remain correct: latest-only evaluation misses these historical failures.

Removing provenance gives common-payload Full/Local exactness of 26/32 versus 32/32 for DS and 11/32 versus 32/32 for Qwen; numerical queries are 62/64 versus 64/64 and 52/64 versus 64/64. Source/time unavailability follows mechanically from nulling those fields, not a novel result. JSON and SQLite independently reload identical content and agree on all 1,024 factorial-study queries: a backend negative control, not an LLM summary or graph-memory baseline.

Table 8: Retained-provenance maintenance and subsequent full-history baseline. Tokens are whole-run provider totals, including writing at every step.

Model	Update	Exact state	Queries	Input	Output
DS	Full	32/32	128/128	40,700	31,181
DS	Local	32/32	128/128	40,380	4,644
DS	History	31/32	128/128	67,356	31,194
Qwen	Full	17/32	107/128	47,979	52,171
Qwen	Local	32/32	128/128	48,171	7,719
Qwen	History	30/32	128/128	74,785	52,751

All calls use direct routes, temperature zero, thinking disabled, 120-second timeout and no retries; returned model IDs match. DS uses parser v2's nonsemantic removal of exact `type:json_object` metadata; original Qwen runs use v1. A separate 128-call DS v1 diagnostic has 127 envelope rejections, four also violating the event schema. It is not pooled with primary results; normalization never repairs state.

Full historical-source rereading. The 64-call History baseline uses parser v2 and more information, not matched retention. All 256 queries pass, yet DS mis-keys events at one checkpoint and Qwen misstates unqueried times at two. Against History, Local reduces input tokens by 40.0%/35.6% and total tokens by 54.3%/56.2% for DS/Qwen. History uses 226,086 tokens (USD 0.057653; CNY 0.571578). This post-DEV comparison is not confirmation or a lifecycle break-even measurement.

Matched correct-prior diagnostic. In 64 additional Qwen calls, both update modes receive identical correct previous-step state and current reports. Generated outputs do not feed the next step. Full/Local achieve 28/32 versus 32/32 exact states and 116/128 versus 128/128 queries. These oracle-assisted resets are not autonomous trajectories. Of 18 prompts identical to the earlier Full run, one yields a different state, preventing attribution of all improvement to reset priors. Record-level auditing also tempers state-level failure counts: persistent Full retains 375/409 correct event checkpoints, controlled Full 404/409, and Local 409/409; these repeated records are not independent samples. Neither a new algorithm nor model-general reliability follows. Hidden-change discovery and query-time applicability reassessment remain untested.

C MATCHED-SOURCE MEMORY REPRESENTATION DIAGNOSTIC

This exploratory comparison tests learned compression and reading, not online maintenance. At checkpoints 8 and 16 of the two fictional sequences in Appendix B, each writer sees all reports available so far and compiles either compact structured rows or a free-text summary. Both know the same six fields, correction semantics and four query types; neither sees query instances, gold states or future reports. Memory is limited to 2,048 or 4,096 UTF-8 bytes. Structured rows use six positional fields, avoiding repeated field names. The same model then answers four queries in one call using an identical prompt template for both representations; only the stored memory differs. Both format legends are supplied to both readers and charged as input tokens. Source access and capacity are matched, not retained information or tokenizer length.

An initial 64-call pilot exposed an interface issue: DeepSeek returned named objects instead of positional rows, rejecting all eight structured writes (six overflows, two schema failures). We preserved this pilot and revised the writer prompts to include the same format-only examples in both arms, explicitly requesting arrays of arrays for structured output. The reader, data, capacities and scorer were unchanged. The subsequent 64-call DEV comparison is reported in Table 9; it is not a held-out confirmation after prompt development. The exact models are `deepseek-flash` and `qwen3.7-plus-2026-05-26`, both through direct routes with the settings used in Appendix B.

Capacity compliance matters. All structured final-snapshot writes exceed 2,048 bytes, versus neither DeepSeek summary and both Qwen summaries. There are no schema rejections in the revised run. Complete canonical positional states occupy approximately 2,468 bytes at the final snapshots: this is a size diagnostic, not oracle information given to writers. At 4,096 bytes, DeepSeek ties at 16/16, while Qwen scores 14/16 versus 11/16. The constrained setting therefore does not establish uniform

Table 9: Revised-prompt comparison. Entries are correct typed-exact answers out of eight per checkpoint and capacity (two families); timestamp answers require exact string equality, not merely the same instant. Rejected writes remain in the denominator and give the reader empty memory; no truncation or semantic repair is performed.

		2,048 bytes		4,096 bytes	
Model	Checkpoint	Structured	Summary	Structured	Summary
DeepSeek	8	8/8	6/8	8/8	8/8
DeepSeek	16	0/8	7/8	8/8	8/8
Qwen	8	7/8	5/8	8/8	5/8
Qwen	16	0/8	0/8	6/8	6/8

structured-memory superiority. Accepted-only scores are selection-conditioned and cannot remove the capacity failures.

Writing and reading also fail differently. A deterministic post-hoc reader answers all 24 queries on accepted Qwen structured memories correctly, whereas the shared LLM reader misses three latest-value queries. This diagnostic is not substituted into the primary comparison. DeepSeek's three summary errors in the revised run are timestamp strings omitting zero seconds; they denote the correct instants. Timestamp-equivalence scoring, reported separately from frozen exact-string scoring, raises its constrained-summary score from 13/16 to 16/16. These errors must not be interpreted as loss of temporal facts.

All 128 calls completed, using 355,538 provider-reported tokens (USD 0.080786; CNY 0.672516), with write/read usage logged separately. Two synthetic families, known queries and independently rebuilt snapshots do not establish unseen-task generalization, summary maintenance or query-time revalidation. Structured format alone guarantees neither factual preservation nor reliable querying.