

# Skill as Memory, Not Document: A Database-Native Substrate for Agent Skill Catalogs

Pranab Sarkar, Independent Researcher (ORCID 0009-0009-8683-1481)

2026-05-11

## Contents

<b>1</b>	<b>Skill as Memory, Not Document: A Database-Native Substrate for Agent Skill Catalogs</b>	<b>2</b>
1.1	Abstract . . . . .	2
1.2	1. Introduction . . . . .	3
1.2.1	1.1 Contributions . . . . .	4
1.2.2	1.2 What this paper deliberately does NOT claim . . . . .	4
1.3	2. The Category Error . . . . .	5
1.3.1	2.1 Documents and memory: different optimization criteria . . . . .	5
1.3.2	2.2 What “human-readable structure” costs in agent memory . . . . .	5
1.3.3	2.3 Three failure modes as symptoms of the category error . . . . .	5
1.3.4	2.4 An operational definition of “memory” . . . . .	6
1.4	3. Three Failure Modes — Formal Characterization . . . . .	6
1.5	4. The Substrate — Memory-Shaped Storage . . . . .	7
1.5.1	4.1 Five operational invariants . . . . .	7
1.5.2	4.2 API surface . . . . .	7
1.5.3	4.3 Implementation . . . . .	7
1.5.4	4.4 What the substrate is NOT . . . . .	8
1.6	5. Failure Mode F1: Token Burn at Scale . . . . .	8
1.6.1	5.1 Setup . . . . .	8
1.6.2	5.2 Five disclosure patterns . . . . .	8
1.6.3	5.3 Kill-test ablation — where does the 1.489× come from? . . . . .	10
1.6.4	5.4 K-sensitivity (cl100k_base) . . . . .	10
1.6.5	5.5 Multi-tokenizer validation (K=5) . . . . .	10
1.6.6	5.6 Body-length sensitivity . . . . .	11
1.7	6. Failure Mode F2: Slowdown at Scale . . . . .	11
1.7.1	6.1 Setup . . . . .	11
1.7.2	6.2 Latency . . . . .	11
1.7.3	6.3 Retrieval quality . . . . .	12
1.7.4	6.4 Scope-boundary . . . . .	12
1.8	7. Failure Mode F3: Hallucination Admission . . . . .	12
1.8.1	7.1 Adversarial corpus . . . . .	12
1.8.2	7.2 Substrates compared . . . . .	12
1.8.3	7.3 Headline results . . . . .	12
1.8.4	7.4 Per-category breakdown . . . . .	13

1.8.5	7.5 Interpretation and scope-boundary . . . . .	13
1.9	8. Operational Decomposition — What a Competent Filesystem Alternative Must Assemble	14
1.10	9. Autonomous-Learning Consumers — Three Integration Observations . . . . .	14
1.10.1	9.1 WisePick (capability routing over outcome history) . . . . .	15
1.10.2	9.2 Hermes Plugin (filesystem-vs-runtime skill lifecycle) . . . . .	15
1.10.3	9.3 RAI AI (state-recovery-aware routing) . . . . .	15
1.10.4	9.4 What these support . . . . .	15
1.11	10. Related Work . . . . .	15
1.11.1	10.1 Filesystem-based skill catalogs (document-shape) . . . . .	15
1.11.2	10.2 Vector-store-backed memory frameworks . . . . .	15
1.11.3	10.3 Database-backed agent infrastructure . . . . .	16
1.11.4	10.4 Schema and capability semantics . . . . .	16
1.11.5	10.5 Skill systematizations and benchmarks . . . . .	16
1.11.6	10.6 Position . . . . .	16
1.12	11. Methodology Notes . . . . .	16
1.13	13. Limitations and Falsification Conditions . . . . .	17
1.13.1	13.1 What I did not measure . . . . .	17
1.13.2	13.2 Falsification conditions . . . . .	17
1.13.3	13.3 AGPL licensing tradeoff . . . . .	18
1.14	14. Discussion . . . . .	18
1.14.1	14.1 What the category framing implies . . . . .	18
1.14.2	14.2 When the framing matters . . . . .	18
1.15	15. Future Work . . . . .	18
1.16	16. Conclusion . . . . .	19
1.17	References . . . . .	19
1.17.1	Primary system citation . . . . .	19
1.17.2	Skill systems and infrastructure . . . . .	19
1.17.3	Skill-related research . . . . .	20
1.18	Appendix A — Reproducibility . . . . .	20

# 1 Skill as Memory, Not Document: A Database-Native Substrate for Agent Skill Catalogs

**Author:** Pranab Sarkar, Independent Researcher **ORCID:** 0009-0009-8683-1481 **Date:** 2026-05-11 **License:** CC BY 4.0 **DOI:** 10.5281/zenodo.20128887 **Related software:** YantrikDB — <https://doi.org/10.5281/zenodo.18793952>  
**Code & data:** [https://github.com/yantrikos/yantrikdb-server/tree/main/benchmarks/skill\\_recall](https://github.com/yantrikos/yantrikdb-server/tree/main/benchmarks/skill_recall)

## 1.1 Abstract

Current LLM agent skill systems use storage formats optimized for human editorial workflow: Anthropic Agent Skills stores skills as SKILL.md files with YAML frontmatter and Markdown body; Voyager persists agent-authored skills as Python files. These document-first conventions collapse human authoring format, retrieval metadata, and agent-consumed runtime body into a single artifact. As autonomous agent learning becomes a primary use case — skills written by agents at runtime, retrieved by agents at inference, scoring composed over agent-emitted outcome events — document-shape choices become operational baggage.

I frame the required shift as **“skill as memory, not document”** and characterize three recurring failure modes when document-first catalogs are used as agent memory substrates: (i) **token burn** from in-context structural metadata; (ii) **slowdown** from filesystem organization not optimized for retrieval; (iii) **invalid-skill admission** from YAML-parseable acceptance.

I characterize these failure modes formally and measure all three on a 5,000-skill corpus. I report a database-native resolution implemented as YantrikDB v0.8.13, an open-source skill substrate that stores skills as agent-shaped memory: structural metadata as indexed database columns (projected out of agent-consumed content), body as opaque-to-humans agent-optimized prose, outcome events as append-only raw signal with no server-side aggregation. (i) Token burn: full-catalog disclosure at 5,000 skills consumes 919,200 tokens, exceeding GPT-4 Turbo 128K and Claude 3.7 200K windows entirely; the substrate’s top-K disclosure consumes 369 tokens, constant in catalog size. A controlled ablation attributes the per-query gap against optimized filesystem alternatives to ~36 tokens of YAML frontmatter overhead per retrieved skill — small but architecturally clean. (ii) Slowdown: p50 retrieval latency 87.3 ms, p95 106.3 ms at 5,000-skill scale on single-node deployment. (iii) Invalid-skill admission: substrate rejects 70 of 70 adversarially-malformed skills (0% admission rate) at write time; a document-only YAML-parseable admission policy baseline admits 68 of 70 (97% admission).

The contributions are: the **“skill as memory, not document” framing** as a unifying lens for autonomous-learning substrate design; the **three-failure-mode taxonomy** and its empirical measurement at 5,000-skill scale on a public reproducible corpus; and an **operational decomposition** of substrate-level primitives a competent filesystem alternative would need to assemble. I do not claim YantrikDB is a new database architecture — all individual primitives it ships (typed records, vector index, append-only logs, schema validation, Raft replication) exist independently in prior work. I identify a category mismatch in document-first skill catalogs and demonstrate a database-native design pattern for agent-written skill memory. I document three independent integrators, state explicit falsification conditions, and pre-specify a follow-up evaluation against indexed Postgres+pgvector boring baselines and end-to-end agent task success.

**Keywords:** autonomous agent learning, agent memory, skill management, vector retrieval, database substrate, schema validation, context window, experience report.

---

## 1.2 1. Introduction

The dominant patterns for LLM agent skill management — Anthropic’s Agent Skills specification [Anthropic, 2025], Voyager’s `skill_library/` directory [Wang et al., NeurIPS 2023], framework-level tool registries — treat skills as **documents**: files on disk with human-readable structure, optimized for human editorial workflow. SKILL.md files have YAML frontmatter that humans read; Markdown bodies are formatted for IDE display and Git diff. Voyager’s Python skill files are code humans can debug. These format choices presuppose that the skill catalog is **curated** — humans author, review, version, and edit.

Autonomous agent learning at scale changes this presupposition. When agents write skills at runtime via define-style APIs, when catalogs grow past ~1,000 skills, when production systems demand audit logs and tenant isolation, the human-editorial assumptions become operational baggage. I argue this is a category error: **filesystem skill catalogs are documents; what autonomous learning needs is memory**.

Memory and documents have different optimization criteria. Documents optimize for human readability (section structure, YAML frontmatter), human authoring workflow (Git-friendly file organization, IDE editability), and human review (PR diffs, code review). Memory optimizes for retrieval cost (index-bound

disclosure, projected-out metadata), write safety (schema enforcement at the API boundary), and machine consumption (compact body content, no human-readable wrappers).

When a skill catalog is consumed by humans (editing a SKILL.md in an IDE), the document format is right. When consumed by agents at runtime (retrieving top-K candidates for a query, scoring outcome history), memory shape is right. As autonomous learning becomes the primary consumer, the substrate should change shape.

This paper makes that argument concrete. I characterize three failure modes that arise when document-shaped substrates are used as agent memory at scale (Section 3); I describe a database-native substrate (YantrikDB) that ships memory-shaped storage with the required operational primitives (Section 4); I measure each failure mode on a 5,000-skill corpus (Sections 5–7); I enumerate the operational primitives a competent filesystem alternative would need to assemble (Section 8); I document three independent integrators using the substrate for autonomous learning (Section 9); and I state falsification conditions and pre-specify a follow-up evaluation (Sections 12–14).

### 1.2.1 1.1 Contributions

1. **The “skill as memory, not document” framing** as a unifying lens for autonomous-learning substrate design. Prior work either accepts the filesystem-as-substrate choice without examination (Anthropic Agent Skills spec) or persists skills to files because that was the artifact-level abstraction (Voyager). I do not claim YantrikDB is a new database architecture; I identify a substrate mismatch in document-first skill catalogs and demonstrate a database-native design pattern for agent-written skill memory.
2. **A three-failure-mode taxonomy** (F1/F2/F3) — recurring substrate-level symptoms that appear when document-first catalogs are used as agent memory, with formal thresholds (Section 3).
3. **Empirical measurements** of all three failure modes at 5,000-skill scale on a public reproducible corpus (Sections 5–7).
4. **An operational decomposition** (Section 8) of substrate-level primitives a competent filesystem alternative must assemble to close F1+F2+F3 simultaneously.

### 1.2.2 1.2 What this paper deliberately does NOT claim

- **Not a paradigm shift.** Individual primitives (typed records, vector index, append-only logs, Raft, schema validation) exist independently in prior database, IR, and event-sourcing literature. The contribution is the framing + the operational packaging argument, not a new primitive.
- **Not that filesystems are inherently incapable** of solving the problem. A filesystem-backed system that adds external indexing, write-time validation, transactional updates, and append-only outcome logs becomes — by construction — an agent memory substrate. The question is which substrate ships those primitives together.
- **Not novel against an indexed Postgres+pgvector + JSON-schema + audit-table assembly.** That assembly could close all three failure modes; pre-specified as the most important deferred comparison.
- **Not end-to-end agent task success.** Retrieval cost / latency / admission rates are reported; downstream task accuracy is pre-specified for the follow-up paper.
- **Not large-scale production attestation.** Single-node homelab + three early integrators. Multi-tenant contention, leader failover, concurrent-writer behavior at 50K+ scale all deferred.

## 1.3 2. The Category Error

### 1.3.1 2.1 Documents and memory: different optimization criteria

Optimization axis	Documents (human-curated)	Memory (agent-consumed)
Primary reader	Humans	Agents
Authoring workflow	IDE, Git, code review	Runtime API, no human in loop
Structure	Section headers, frontmatter, formatting	Whatever the retrieval/consumption pipeline needs
Update cadence	Days/weeks (PRs)	Milliseconds (per-execution outcome events)
Consistency model	Eventual (Git merge conflicts ok)	Strong (transactional, ACID-style)
Validation	Code review, optional CI	Required at write API boundary
Audit	Git log	Append-only typed event log
Multi-tenant	File-system ACLs (OS-level)	API-level tenancy enforcement
Replication	Git remotes, NFS, manual sync	Consensus protocol (Raft)

Filesystem skill catalogs use documents-optimized choices in nearly every row. That is right when humans are the primary author/reader. It becomes operational baggage when agents are. The point is not that filesystems cannot be re-engineered into agent memory substrates — they can, by adding the missing primitives — but that the document-first defaults need to be replaced rather than extended.

### 1.3.2 2.2 What “human-readable structure” costs in agent memory

The YAML frontmatter on a SKILL.md file (`name:`, `applies_to:`, `triggers:`, `skill_type:`) exists because humans editing the file need to see and change those fields. When the same content is retrieved into an LLM context at runtime:

- The structural metadata duplicates information already in the retrieval index (`applies_to` was used to filter retrieval; the agent doesn’t need to see it again).
- The YAML formatting (`---`, `name:`, indented lists) consumes prompt tokens that the agent does not act on.
- The structural markers (`## Procedure`, `## Examples`) imply distinct semantic sections, which can pull agent attention into reasoning modes (e.g., “which section is relevant here?”) that retrieval has already resolved.

A memory-shape storage projects all of this out: the structural metadata is queryable as indexed columns server-side but never enters the agent-consumed body. The body itself is what the agent actually needs — and only that.

### 1.3.3 2.3 Three failure modes as symptoms of the category error

The category error manifests as three coupled failure modes at scale. I define them formally in Section 3 and measure them empirically in Sections 5–7:

- **(F1) Token burn** — symptoms of “structural metadata that exists for humans is dragged into agent prompts.”
- **(F2) Slowdown** — symptoms of “filesystem organization optimized for Git is not retrieval-optimized.”

- **(F3) Invalid-skill admission** — symptoms of “YAML-parseable acceptance optimized for human authoring lets agents author junk.”

These are not independent design oversights; they share a common cause — the same human-editorial defaults that make document formats good for curation make them costly for runtime agent consumption. Closing them simultaneously requires either replacing document-first defaults at the substrate layer (as the database-native approach does) or layering enough additional infrastructure on top of the filesystem that the underlying file format is no longer the canonical agent-consumed artifact. I catalog what such additional infrastructure must include in Section 8.

### 1.3.4 2.4 An operational definition of “memory”

I use “memory” in this paper with a specific operational meaning. By a **skill memory substrate** I mean a runtime substrate that provides:

- **Projected agent-consumed bodies** — structural metadata stored as queryable database columns rather than in-text frontmatter
- **Indexed retrieval** — top-K disclosure cost bounded by K, not by catalog size N
- **Write-time admission control** — schema validation enforced at the API boundary, not as an optional downstream check
- **Append-only outcome histories** — typed events with no server-side aggregation, preserving raw signal for consumer-side scoring

This is not a metaphorical use of “memory.” It is a specific set of substrate-layer guarantees. A system satisfying these guarantees is in the memory category regardless of underlying storage (filesystem, database, custom binary format); a system not satisfying them is in the document category regardless of the same. The four properties define the category; YantrikDB (Section 4) is one implementation.

---

## 1.4 3. Three Failure Modes — Formal Characterization

For a skill substrate of size N exposing API endpoints {define, search, outcome, forget}:

**(F1) Token burn at scale.** Let  $T_{\text{disclose}}(N, K)$  be the prompt-token cost of disclosing K relevant skills from a catalog of N to the model. F1 is violated when  $T_{\text{disclose}}$  grows with N rather than K. F1 binds when  $T_{\text{disclose}}(N, K) > T_{\text{window}}$  for production context windows. At current limits (Claude 3.7: 200K, GPT-4 Turbo: 128K), naive in-context catalog disclosure violates F1 at  $N \approx 1,000\text{--}4,000$  depending on body length.

**(F2) Slowdown at scale.** Let  $L_{\text{query}}(N)$  be the wall-clock latency of returning top-K relevant skills from a catalog of N. F2 is violated when  $L_{\text{query}}$  grows with N at production usage rates. Filesystem-walk approaches pay  $L_{\text{query}}$  proportional to N. Index-bound retrieval (HNSW, IVF, BM25) achieves  $O(\log N)$  or  $O(\text{constant})$  for fixed K, but the index must be built (cold start), maintained (write invalidation), and kept consistent (transactional ordering) — operational properties filesystem-as-substrate does not provide natively.

**(F3) Invalid-skill admission.** Let  $A_{\text{malformed}}$  be the fraction of structurally-malformed skill definitions that the substrate admits at write time. F3 is violated when  $A_{\text{malformed}} > 0$  — i.e., when the substrate’s API boundary does not enforce structural validity. Filesystem-backed substrates that validate only YAML-parseability admit shape-violating skills (Section 7 measures  $A_{\text{malformed}} \approx 97\%$  against an adversarial

corpus). Schema-validating substrates achieve  $A_{\text{malformed}} = 0\%$  against structural violations; semantic violations remain agent-layer responsibility.

**Coupling.** A vector-index layer that closes F2 still admits hallucinated skills if F3 is unenforced at the same boundary. A CI-time validator that closes F3 still pays F2 cold-start parse cost. A cache for F1 still serves bad-shape skills if F3 is unenforced. The three primitives required to close F1+F2+F3 simultaneously must operate at the same API boundary, transactional with each other.

---

## 1.5 4. The Substrate — Memory-Shaped Storage

I describe YantrikDB as one existence proof that an agent-memory substrate addressing the evaluated forms of all three failure modes is feasible as an open-source single-binary deployment. I do not claim it is the only such proof; any system satisfying the four operational definitions of Section 2.4 would qualify.

### 1.5.1 4.1 Five operational invariants

1. **Skill identity invariant.** Skill = first-class typed database entity. `skill_id` matches `^[a-z][a-z0-9_]*(\.[a-z0-9_]+)$`, length 4–200. `skill_type` is one of {procedure, reference, lesson, pattern, rule}. `applies_to` 1–10 entries matching `^[a-z][a-z0-9_]*$`. body length 50–5000. Enforced server-side at write (closes F3).
2. **Disclosure invariant.** Runtime LLM prompt disclosure bounded by top-K of `POST /v1/skills/search`. Structural metadata (`skill_id`, `skill_type`, `applies_to`, `triggers`) stored as indexed database columns, **projected out of agent-consumed content** (closes F1).
3. **Outcome provenance invariant.** `POST /v1/skills/{id}/outcome` appends typed events to an append-only audit log. Substrate does NOT summarize or roll up. Aggregation is consumer’s responsibility. Operational consequence: substrate-level audit trail for autonomous-learning evaluation; no derived state contamination.
4. **Runtime authoring invariant.** Agents author skills via the same validated API as humans. Substrate distinguishes `authored_by` but not `who-may-author`. Schema validation applies identically — this is the F3-closure mechanism for agent-generated content.
5. **Replication invariant.** Skill definitions and outcome events replicate via deterministic Raft log (open-raft v0.9.x) with per-database tenant boundaries.

### 1.5.2 4.2 API surface

<code>POST /v1/skills/define</code>	Register new skill.	201 / 400 / 409.
<code>GET /v1/skills/{skill_id}</code>	Exact lookup.	
<code>POST /v1/skills/search</code>	Semantic search + filter.	
<code>POST /v1/skills/{skill_id}/outcome</code>	Append outcome event.	
<code>POST /v1/skills/{skill_id}/forget</code>	Tombstone + optional cascade.	

### 1.5.3 4.3 Implementation

- **Storage:** SQLite WAL + HNSW vector index (384-dim default). Indexed columns: `skill_id`, `skill_type`, `applies_to`, `tenant_id`, `created_at`. Body is opaque-to-humans semantic content.

- **Replication:** openraft 0.9.x, per-tenant log isolation, mTLS cluster transport.
- **Schema validation:** server-side regex + length + enum (mirrored verbatim in Section 7 benchmark).
- **Outcome events:** append-only typed records, separate substrate namespace, no auto-rollup.

#### 1.5.4 4.4 What the substrate is NOT

- Not a tool registry. Skills are descriptive procedural knowledge, not executable function schemas.
- Not a workflow engine. Substrate stores; consumers invoke.
- Not a capability ontology. `applies_to` is free-form tag array (regex-validated, not controlled-vocabulary-validated).
- Not an evaluation framework. Outcomes are events; consumer decides success/failure semantics.

### 1.6 5. Failure Mode F1: Token Burn at Scale

I measure prompt-token cost under five disclosure patterns on the 5,000-skill corpus.

#### 1.6.1 5.1 Setup

- **Corpus:** 5,000 synthetic skills, deterministic seed=42, 50 topic-families × 10 action-types × 10 variants. Median body 123 tokens (cl100k\_base). Public.
- **Queries:** 100 ground-truth queries with target `skill_id` lists.
- **Tokenizers:** cl100k\_base primary; o200k\_base cross-validation.
- **Top-K:** 5 unless otherwise stated.

#### 1.6.2 5.2 Five disclosure patterns

Catalog N	(A) Full catalog in prompt	(B) Naive progressive disclosure	(C) Indexed filesystem, frontmatter rendered	(D) Indexed filesystem, frontmatter parsed for index, body-only in prompt	(E) YantrikDB top-K body
100	18,250	5,388	549	369	369
500	91,450	25,268	549	369	369
1,000	181,200	49,868	549	369	369
2,500	455,150	124,768	549	369	369
5,000	919,200	252,868	549	369	369

(A) at N=5,000 exceeds GPT-4 Turbo 128K and Claude 3.7 200K context windows. (B) at N=5,000 exceeds Claude 200K. (C), (D), (E) are constant-cost in N at fixed K — index-bound disclosure. The “exceeds context window” framing is specific to the cited models; production deployments using Gemini 1.5 Pro (1M tokens) or other extended-context models may not hit a hard ceiling at this corpus size. The substantive concern remains: prompt-disclosure cost growing linearly with catalog size compounds with every other prompt component (system prompt, conversation history, tool definitions, output budget) and degrades attention-quality regardless of nominal context limit.

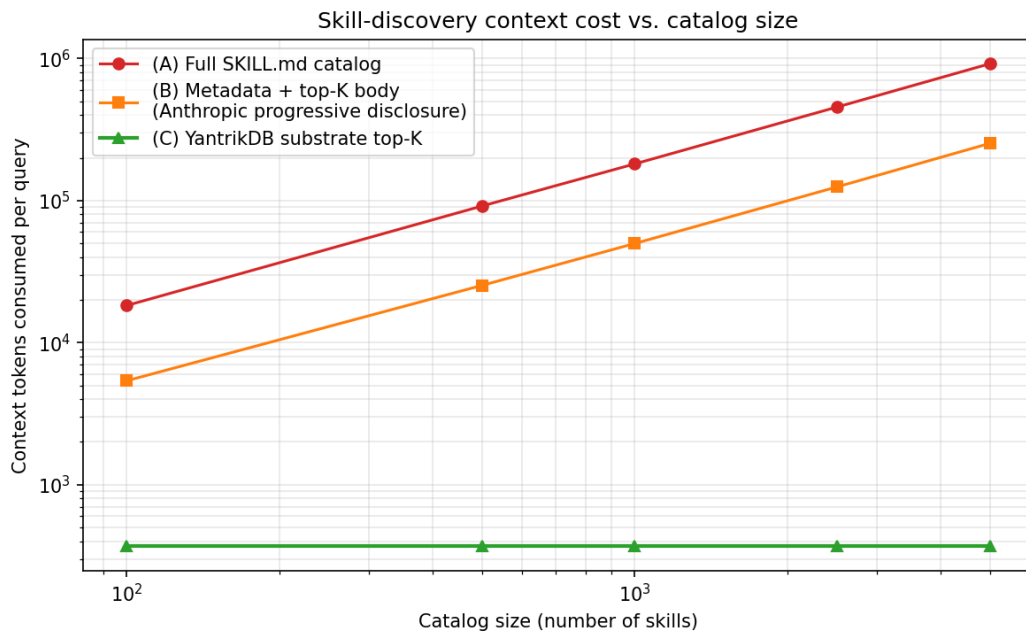


Figure 1: Figure 1: Per-query top-K=5 prompt-token cost vs catalog size, log-log axes. Patterns (A) full-catalog and (B) naive progressive disclosure scale linearly with N and exceed production context windows past  $\sim 1K$  skills. Patterns (C), (D), (E) are constant in N. (D) and (E) coincide exactly because they tokenize the same body text after the kill-test ablation strips YAML frontmatter from (C). Source: `benchmarks/skill_recall/token_cost_bench.py`.

The relevant comparison for autonomous learning at scale is among the index-bound patterns. **(D)  $\equiv$  (E)** at all percentiles. The 1.489 $\times$  ratio (C)/(E) is **entirely YAML frontmatter overhead** — approximately 36 tokens per retrieved skill.

### 1.6.3 5.3 Kill-test ablation — where does the 1.489 $\times$ come from?

Representation	mean	p95
(C) SKILL.md with frontmatter	549	949
(D) SKILL.md body-only (frontmatter parsed for index, omitted from prompt)	<b>369</b>	<b>635</b>
(E) YantrikDB top-K body	<b>369</b>	<b>635</b>

(D) and (E) are identical at every percentile because they tokenize the same body text. **The architectural attribution is clean: filesystem substrates that ship YAML frontmatter as in-text content pay an overhead that database substrates avoid by storing metadata as indexed columns. An indexed filesystem implementation that strips frontmatter post-retrieval reaches per-query parity with the database substrate.**

I do not claim this 1.489 $\times$  is a substrate advantage in the architectural sense. It is a constant-factor consequence of storing skill metadata as agent-consumed text vs as indexed columns. The substantive F1-closure claim is independent: **any** index-bound disclosure pattern (C, D, or E) closes F1 at fixed K; document-shape patterns (A, B) do not.

### 1.6.4 5.4 K-sensitivity (cl100k\_base)

K	(E) YantrikDB mean	(C) SKILL.md mean	Ratio
1	122	181	1.487 $\times$
3	244	364	1.489 $\times$
5	369	549	1.489 $\times$
10	679	1,011	1.489 $\times$
20*	679	1,011	1.489 $\times$

\*K=20 saturates against the corpus (each query has 10 target-family variants).

The ratio is stable across K because the frontmatter overhead is per-skill, not per-query.

### 1.6.5 5.5 Multi-tokenizer validation (K=5)

Tokenizer	(C) mean	(E) mean	Ratio
cl100k_base (GPT-4 family)	549	369	1.489 $\times$
o200k_base (GPT-4o / 4.1 family)	547	366	1.495 $\times$

Ratio stable to within 0.4% between these two OpenAI tokenizers. Stability across Anthropic, Llama, or other tokenizer families is not empirically verified here; the YAML-frontmatter overhead is structurally identical across tokenizers under the analytical decomposition, but reported numbers above are specific to cl100k\_base and o200k\_base measurements.

### 1.6.6 5.6 Body-length sensitivity

The ratio is body-length-dependent. As body length grows, fixed ~36-token frontmatter is a smaller fraction:

Body tokens/skill	Ratio (C)/(E)
75	1.48×
123 (this corpus)	1.49×
250	1.14×
500	1.07×
1,000	1.04×
2,000	1.02×

In production corpora with longer bodies, the ratio approaches parity. **F1-closure does not depend on the ratio**: index-bound disclosure (C/D/E) closes F1 regardless.

## 1.7 6. Failure Mode F2: Slowdown at Scale

Pre-existing measurement at commit c886e9e (2026-04-29). **This section establishes bounded prototype retrieval-latency behavior at the evaluated 5,000-skill scale on a single-node deployment. It is not claimed as a superiority result against optimized Postgres+pgvector, SQLite FTS, vector-database, or filesystem-plus-index baselines** — those comparisons are reserved for the follow-up paper (Section 14). The substantive F2 claim of this paper is qualitative: retrieval latency does not grow with catalog size at 5,000 scale under the substrate’s HNSW + indexed-column design.

### 1.7.1 6.1 Setup

- Windows 11 + Docker Desktop, single-node.
- Bundled MiniLM-L6-v2 embedder (384-dim).
- HNSW index (ef=200, M=16 — defaults).
- 100 ground-truth queries from §5.

### 1.7.2 6.2 Latency

Metric	Value
Retrieval p50	87.3 ms
Retrieval p95	106.3 ms
Retrieval max	119.9 ms
Write throughput	26.1/sec, 0 failures

### 1.7.3 6.3 Retrieval quality

Metric	Value
recall@5 overall	0.86
recall@5 family (broad)	0.96
recall@5 variant (sharp)	0.76

Variant recall of 0.76 is the substantive operational caveat — for variant-precision-sensitive consumers (safety-critical), K must increase or post-filtering apply.

### 1.7.4 6.4 Scope-boundary

Single-node, non-production-class environment. Deferred (Section 13): Linux bare-metal, p99, concurrent-query scaling, embedding/search/SQLite-fetch latency breakdown, embedder sensitivity (MiniLM vs BGE vs text-embedding-3-small), behavior at 50K / 500K / 1M scale.

The F2-closure claim at the present scale: retrieval latency does not grow with catalog size at 5,000 skills.

---

## 1.8 7. Failure Mode F3: Hallucination Admission

I measure the fraction of structurally-malformed skill definitions the substrate admits at write time, on a deterministic adversarial-skill corpus.

### 1.8.1 7.1 Adversarial corpus

90 skills: 20 well-formed controls + 70 adversarial entries across 14 violation classes targeting structural rules an LLM might violate during runtime authoring:

- **skill\_id**: hyphens (5), starts-digit (5), uppercase (5), no-dots (5)
- **applies\_to**: hyphens (5), starts-digit (5), uppercase (5), >10 entries (5), empty array (2)
- **skill\_type**: not-in-enum (5), case/bad (5)
- **body**: too-short (5), too-long (5)
- **missing required**: skill\_id (2), skill\_type (2), body (2), skill\_id-not-string (2)

Deterministic. Public.

### 1.8.2 7.2 Substrates compared

- **(YDB)**: Server-side schema validation — regex + length + enum + required-field. Rejects at write with 400-class error.
- **(FS)**: Naive SKILL.md filesystem — accepts any YAML-parseable input. Field-shape errors surface only at read/use.

### 1.8.3 7.3 Headline results

Subset	N	YDB admitted	FS admitted
Well-formed controls	20	<b>20 (100%)</b>	<b>20 (100%)</b>
<b>Adversarial</b>	<b>70</b>	<b>0 (0%)</b>	<b>68 (97%)</b>

Base-rate admission is identical for valid input. Adversarial admission diverges by 97 percentage points.

The 68/70 admission by the document-only baseline is expected — the baseline by definition has no validator beyond YAML-parseability. This result should not be read as showing that filesystems are inherently incapable of enforcing such constraints. It shows that **a filesystem with only YAML-parseability as admission control does not enforce them**; a filesystem with an external CI-time validator, pre-commit hook, or write-API wrapper could reduce admission rate. The substantive point is that such a validator must be built and operated separately. The substrate ships the validator as a property of the storage layer.

#### 1.8.4 7.4 Per-category breakdown

Class	N	YDB	FS
skill_id_hyphen	5	0	5
skill_id_starts_digit	5	0	5
skill_id_uppercase	5	0	5
skill_id_no_dots	5	0	5
applies_to_hyphen	5	0	5
applies_to_starts_digit	5	0	5
applies_to_uppercase	5	0	5
applies_to_too_many	5	0	5
applies_to_empty	2	0	2
skill_type_not_in_enum	5	0	5
skill_type_case_or_bad	5	0	5
body_too_short	5	0	5
body_too_long	5	0	5
missing_body	2	0	2
missing_skill_type	2	0	2
skill_id_not_string	2	0	2
missing_skill_id	2	0	0

Every category the filesystem admits passes through downstream to agent invocation, failing late.

#### 1.8.5 7.5 Interpretation and scope-boundary

The substrate’s write-time schema validation closes F3 for **structural** violations: agents authoring via POST /v1/skills/define cannot poison the substrate with shape-violating definitions. The substrate does NOT catch **semantic** violations (well-formed-but-dangerous content) — this is the deliberate scope-boundary of schema-not-semantics.

The document-only YAML-parseable admission policy baseline is intentionally weak (YAML-parseable acceptance). A SKILL.md system with CI-time or pre-write validation can reduce admission rate. **The substantive**

**point:** such a validator must be built and operated separately. The substrate ships it as a property of the storage layer.

---

## 1.9 8. Operational Decomposition — What a Competent Filesystem Alternative Must Assemble

Section 5 showed an indexed filesystem alternative (pattern D) reaches per-query token parity. Section 7 showed a CI-time validator can reduce filesystem F3 admission. I enumerate what such a competent filesystem alternative must assemble to close F1 + F2 + F3 simultaneously:

Primitive	What it provides	Failure mode without it
External vector / BM25 index	Index-bound retrieval	F2: linear filesystem scan
YAML parsing pipeline	Index extraction	F2: per-query parse cost
File-change invalidation (inotify / fsevents / RDCW)	Index freshness	F2: stale retrieval after edits
Cold-start indexing	Initial index build	F2: multi-second first-query latency
Transactional update	Consistency under concurrent edit/read	F3: torn parse on race
Write-time schema validator	Shape enforcement	F3: malformed skills admitted
Multi-tenant isolation	Tenant boundary	Cross-tenant skill leakage
Outcome event log database	Audit trail	No provenance
Concurrent writer serialization	Atomic write	F3: duplicate skill_id state undefined
Replication for HA	Single-node failure tolerance	SPOF

**Ten primitives.** Each independently solvable; each requires operational engineering. The substrate ships them as a single binary with one consistency model.

I do not claim novelty in any individual primitive. I claim that the **conjunction is the operationally useful unit** for autonomous learning at scale. The operational asymmetry between “one substrate” and “ten assembled services” is measurable in deployment complexity, integration LOC, maintenance burden, and consistency-bug surface area.

The strongest fair comparison is not against the document-only YAML-parseable admission policy baseline measured here but against a **competent Postgres+pgvector+JSON-schema+audit-table assembly**. I pre-specify that comparison as the central deferred measurement (Section 13).

---

## 1.10 9. Autonomous-Learning Consumers — Three Integration Observations

I document three independent adopters using the substrate for autonomous-learning use cases. These are observations, not validation.

### 1.10.1 9.1 WisePick (capability routing over outcome history)

[Wu, 2026, Apache-2.0]. Integrated within ~24 hours of initial design conversation. Mapping: `capability_id` ↔ `skill_id`, `ECU_type` ↔ `skill_type`, “Decision Memory” ↔ append-only outcome event log. Production live as v0.1.2 with `k8s_reconcile` capability mapping. **Pre-flight scoring composes client-side over outcome\_substrate event history — no server-side hook required.** Demonstrates the append-only-with-deferred-aggregation design choice in practice: the substrate provides raw signal, the consumer (WisePick) computes its own scoring function over it.

### 1.10.2 9.2 Hermes Plugin (filesystem-vs-runtime skill lifecycle)

`yantrikdb-hermes-plugin v0.3.0`, design accepted (4 swarm-channel exchanges informed the design; not yet in production). Distinguishes human-authored filesystem skills (durable, version-controlled — *document* shape) from agent-authored YantrikDB skills (runtime-evolving, semantic-searchable — *memory* shape). Both coexist via `metadata.source=hermes` filterability. **Demonstrates the category distinction in practice:** the same plugin maintains documents for human-curated skills AND memory for agent-authored skills, with the substrate boundary clear.

### 1.10.3 9.3 RAI AI (state-recovery-aware routing)

Raised a public state-recovery question that informed the operational specification of `/v1/health.replication_lag_log` for downstream routing. No integration yet. Demonstrates that **substrate observability primitives** matter for autonomous-learning consumers even before full integration.

### 1.10.4 9.4 What these support

Three different agent-layer pedagogies (ECU routing, lifecycle-distinguished authoring, state-recovery-aware routing) compose on the same substrate primitives. **The category distinction (memory vs document) holds in practice: WisePick treats skills purely as memory; Hermes maintains documents and memory separately.** I document this as a design-pattern observation, not validation — task-success deltas and integration-LOC comparisons against baselines are deferred.

---

## 1.11 10. Related Work

### 1.11.1 10.1 Filesystem-based skill catalogs (document-shape)

- **Anthropic Agent Skills** [2025] — `SKILL.md` files with YAML frontmatter and Markdown body; “progressive disclosure” pattern.
- **Voyager** [Wang et al., NeurIPS 2023] — agent-authored Python skill files persisted to disk, retrieved via embedding similarity.

### 1.11.2 10.2 Vector-store-backed memory frameworks

- **mem0, MemGPT/Letta** [Packer et al., 2023], **agent-zero** — skills as memory records in vector indexes (FAISS, pgvector, Chroma, Qdrant). The framework, not the substrate, enforces skill semantics.

### 1.11.3 10.3 Database-backed agent infrastructure

- **Oracle AI Agent Memory** — enterprise database-backed; procedural-memory API surface not publicly documented.
- **DataJoint 2.0** [arxiv 2602.16585] — computational substrate with introspectable schema.
- **AgentTrace** [arxiv 2602.10133] — structured logging.

### 1.11.4 10.4 Schema and capability semantics

- **Beyond Formal Semantics for Capabilities and Skills** [arxiv 2506.11180] — argues against explicit semantic modeling for LLM-based manufacturing automation. Adjacent thesis in a different domain.
- **From Skill Text to Skill Structure** [arxiv 2604.24026] — schema-based validation for skill artifacts.

### 1.11.5 10.5 Skill systematizations and benchmarks

- **SoK: Agentic Skills** [arxiv 2602.20867] — identifies seven design patterns.
- **Externalization in LLM Agents** [arxiv 2604.08224] — review of memory, skills, protocols, harness.
- **SkillRet** [arxiv 2605.05726] — skill retrieval benchmark.
- **Graph of Skills** [arxiv 2604.05333] — dependency-aware structural retrieval over offline skill graphs.
- **EvoSkills** [arxiv 2604.01687], **SkillFoundry** [arxiv 2604.03964], **RL Self-Improving Agent with Skill Library** [arxiv 2512.17102] — self-evolving libraries.

### 1.11.6 10.6 Position

The category-error framing (“skill as memory, not document”) is, to our knowledge, novel in published work. Voyager persists skills to files without examining the choice; Anthropic positions filesystem as the format without comparing to memory-shape alternatives; SoK papers and reviews catalog the field but do not draw the document/memory distinction I propose. I do not claim novelty in any individual primitive (typed records, vector index, append-only logs, schema validation, Raft replication — all exist independently) but in the conceptual framing that unifies their selection.

---

## 1.12 11. Methodology Notes

The empirical claims were developed under adversarial scrutiny. Two-model redteam (OpenAI gpt-5.5 + DeepSeek deepseek-chat, three rounds + synthesizer + a follow-up quick-check) applied during drafting. Key corrections:

- An initial **686× headline ratio** compared YantrikDB against an unrealistic in-context-manifest baseline. After correction to the honest indexed baseline: **1.489×**. The 686× number appears only as the (A)/(E) ratio — anti-pattern reference, not contribution.
- An initial “**schema-not-semantics**” framing was retired due to overlap with arxiv 2506.11180.
- The **kill-test ablation** (Section 5.3) was added after the redteam argued the 1.489× might be representation efficiency. Result attributes the gap to frontmatter overhead per-skill.
- The **invalid-skill admission benchmark** (Section 7) was added in response to the question: “if filesystem catalogs scale to 1000+ files, how do they handle malformed-skill admission?”
- The **three-failure-mode taxonomy** (Section 3) emerged as the unifying frame from the same redteam exchange.

- The **category-error framing** (“skill as memory, not document”) emerged from a post-redteam observation by the author: that the three failure modes share a common cause — using human-editorial formats as agent memory substrates. This is the paper’s thesis-level contribution.

The redteam transcript is preserved in the project repository for methodological auditability.

## 1.13 13. Limitations and Falsification Conditions

### 1.13.1 13.1 What I did not measure

- **Postgres+pgvector+JSON-schema+audit-table boring baseline.** The strongest fair comparison; pre-specified as central deferred measurement.
- **End-to-end agent task success.** Pre-specified.
- **Adversarial near-duplicate retrieval corpus.** The 5,000-skill corpus is regular.
- **Real-world skill corpus.** Lane B production, Anthropic Agent Skills public repo, OpenAPI specs available but not included.
- **Multi-agent contention.** Single-writer throughput only.
- **Long-soak operational tests.** Crash recovery, leader failover, follower bootstrap.
- **Semantic-violation invalid-skill admission.** Schema closes structural; semantic violations are deliberate scope-boundary.
- **Formal security analysis.** Malicious definitions, prompt injection, embedding poisoning.
- **No measurement of downstream prompt-interference or task-success effects** of metadata-rich vs metadata-stripped retrieved content beyond raw token count. Deferred to follow-up paper benchmark 8 (Section 14).

### 1.13.2 13.2 Falsification conditions

The corresponding claim is invalidated if:

1. **(F1’)** Indexed Postgres+pgvector+JSON-schema+audit-table assembly matches the substrate on all three failure-mode measurements with comparable operational complexity.
2. **(F2’)** recall@5 drops below 0.5 on real-world corpora (variant precision required → K must increase → constant-prompt-cost claim weakens).
3. **(F3’)** Real-corpus schema rejection rate exceeds 30% (validation too rigid for production).
4. **(F4’)** Multi-tenant write contention causes replication lag > 30s sustained or audit gaps after crash.
5. **(F5’)** End-to-end task success delta ≤5% vs indexed SKILL.md on fair benchmark.
6. **(F6’)** Outcome event log unusable past 1M events (server-side rollup required, deferred-aggregation claim impractical).
7. **(F7’)** A SKILL.md system with post-retrieval frontmatter stripping AND competent index AND competent validator AND replication AND multi-tenant isolation deploys widely — substrate’s F1+F2+F3 closure becomes literal parity, value rests on packaging only.
8. **(F8’)** LLM-runtime-authored skills measurably lower-quality than human-authored on same task distribution.
9. **(F9’)** A task-success ablation in which agents receive (a) skill content with YAML frontmatter intact vs (b) the same content with frontmatter stripped post-retrieval finds no statistically significant difference in task-success rate, wrong-skill invocation rate, or clarification turns. This would indicate that the substrate’s metadata-projection advantage is purely token-cost and does not affect downstream agent behavior — a finding I would publish in the follow-up paper.

### 1.13.3 13.3 AGPL licensing tradeoff

YantrikDB is AGPL-3.0. Non-trivial barrier for proprietary commercial adoption; appropriate for research-and-substrate positioning. Dual licensing on inquiry. I do not present AGPL as an unambiguous advantage.

---

## 1.14 14. Discussion

### 1.14.1 14.1 What the category framing implies

If the document/memory distinction is right, **partial solutions to autonomous-learning skill management are not solutions**. A vector index over filesystem catalogs closes F1+F2 but leaves F3 open. A CI-time linter closes F3 but does not address F1+F2. A managed memory framework may close F1+F3 but couples skills to a specific agent framework's pedagogy.

The substrate-layer claim is: the three primitives required to close F1+F2+F3 simultaneously — indexed retrieval, write-time validation, append-only provenance — must operate at the same API boundary and within the same consistency domain. The argument is operational, not formal: it is in principle possible to assemble equivalent primitives from independent components (Section 8 enumerates ten), but the assembly costs operational complexity that grows with team size, deployment count, and consistency-bug surface area.

### 1.14.2 14.2 When the framing matters

The category distinction matters most when:

- Skills are agent-authored at runtime, not human-curated.
- Catalogs grow past the regime where in-context disclosure scales (~1,000 skills).
- Audit, replay, and provenance of skill-execution outcomes is operationally important.
- Multiple independent consumers compose pedagogy on shared skill primitives.

The distinction matters less for static catalogs under ~200 skills, single-agent local deployments without audit needs, or use cases requiring formal capability semantics or skill composition (the substrate intentionally declines these).

---

## 1.15 15. Future Work

Follow-up paper pre-specifies the following measurements:

1. **Postgres+pgvector+JSON-schema+audit-table baseline** on the same three failure modes + operational-complexity comparison (LOC, services, configuration burden, integration time).
2. **End-to-end agent task benchmark** with three substrate conditions and task-success metrics.
3. **Adversarial near-duplicate retrieval corpus** measuring recall@K and unsafe-near-miss rate.
4. **Real-world skill subset** (Lane B production, Anthropic Agent Skills public repo, OpenAPI specs, LangChain tool registry).
5. **Multi-agent contention** with 10–100 concurrent writers across tenants.
6. **Threat model and security**: malicious definitions, prompt injection, embedding poisoning.
7. **Semantic-violation invalid-skill admission** beyond structural shape.

8. **Task-success ablation comparing YAML-frontmatter-intact vs frontmatter-stripped skill content** across at least three production LLM providers — measures whether metadata projection affects downstream agent behavior beyond raw token count.

Commitment: follow-up paper within 12 weeks (by 2026-08-04) regardless of outcome.

---

## 1.16 16. Conclusion

I argue that LLM agent skill management at scale faces a category error: filesystem skill catalogs were designed as **documents** for human editorial workflow, but autonomous agent learning at scale needs **memory** for agent consumption. The category error manifests as three coupled failure modes — token burn from in-context structural metadata, slowdown from filesystem organization not optimized for retrieval, invalid-skill admission from YAML-parseable write acceptance. I measure all three at 5,000-skill scale and report a database-native resolution (YantrikDB v0.8.13) that stores skills as agent-shaped memory: structural metadata as indexed columns, body as opaque-to-humans content, outcome events as append-only raw signal.

Headline empirical results: (i) full-catalog disclosure at 5K skills consumes 919,200 tokens, exceeding Claude 200K window; the substrate’s index-bound disclosure consumes 369 tokens, constant in catalog size. The 1.489× margin against an indexed filesystem baseline is YAML frontmatter overhead at ~36 tokens per skill — small but architecturally clean. (ii) p50 retrieval 87.3 ms at 5K-scale single-node. (iii) Substrate rejects 70/70 structurally-malformed skills (0%); document-only YAML-parseable admission policy admits 68/70 (97%).

The paper’s contributions are the **category-error framing**, the **three-failure-mode taxonomy**, the **empirical decomposition** of where the substrate-vs-filesystem token gap actually comes from, the **operational decomposition** of primitives a competent filesystem alternative would need to assemble. I document three autonomous-learning consumers, state explicit falsification conditions, and pre-specify follow-up evaluation against indexed boring-database baselines and end-to-end agent task success.

This is an experience report, not a field-defining paper. The category framing may prove useful as future systems are designed for autonomous learning at scale; the empirical measurements may serve as baselines; the falsification conditions invite challenges that strengthen the field.

---

## 1.17 References

(Full citations to be finalized in formatting pass.)

### 1.17.1 Primary system citation

- Sarkar, P. 2026. *YantrikDB: A Cognitive Memory Database for AI Agents (Software)*. Independent. DOI: 10.5281/zenodo.18793952. Source: <https://github.com/yantrikos/yantrikdb-server>

### 1.17.2 Skill systems and infrastructure

- Anthropic. 2025. *Agent Skills Specification*. <https://platform.claude.com/docs/en/agents-and-tools/agent-skills/overview>
- Wang, G., et al. 2023. *Voyager: An Open-Ended Embodied Agent with Large Language Models*. NeurIPS 2023.

- Wu, J. 2026. *WisePick v0.1.2 — Open-source Apache-2.0 capability-routing toolkit*. <https://github.com/w2jmoe/WisePick>
- Packer, C., et al. 2023. *MemGPT: Towards LLMs as Operating Systems*.
- Rhodes, G. 2026. *Why AI agents need a database for memory, not just a flat file like SKILL.md* (blog post).

### 1.17.3 Skill-related research

- arxiv 2602.20867 — *SoK: Agentic Skills — Beyond Tool Use in LLM Agents*.
- arxiv 2604.08224 — *Externalization in LLM Agents: A Unified Review of Memory, Skills, Protocols and Harness Engineering*.
- arxiv 2506.11180 — *Beyond Formal Semantics for Capabilities and Skills: Model Context Protocol in Manufacturing*.
- arxiv 2604.24026 — *From Skill Text to Skill Structure*.
- arxiv 2604.05333 — *Graph of Skills: Dependency-Aware Structural Retrieval for Massive Agent Skills*.
- arxiv 2605.05726 — *SkillRet: A Large-Scale Benchmark for Skill Retrieval in LLM Agents*.
- arxiv 2602.10133 — *AgentTrace: A Structured Logging Framework for Agent System Observability*.
- arxiv 2604.16911 — *Skilldex: A Package Manager and Registry for Agent Skill Packages*.
- arxiv 2604.01687 — *EvoSkills: Self-Evolving Agent Skills via Co-Evolutionary Verification*.
- arxiv 2604.03964 — *SkillFoundry: Building Self-Evolving Agent Skill Libraries*.
- arxiv 2512.17102 — *Reinforcement Learning for Self-Improving Agent with Skill Library*.

---

## 1.18 Appendix A — Reproducibility

All benchmarks reproducible from `yantrikos/yantrikdb-server/benchmarks/skill_recall/`:

- `generate.py` — corpus generator (deterministic seed=42)
- `token_cost_bench.py` — §5 disclosure-pattern sweep
- `kill_test_ablation.py` — §5.3 body-only ablation
- `k_sensitivity_and_tokenizer_bench.py` — §5.4–5.5
- `invalid-skill_admission_admission_bench.py` — §7 adversarial-corpus measurement

§6 latency from commit `c886e9e` (2026-04-29) per `bench.py` + `RESULTS.md`.

The full GPT-5.5 + DeepSeek redteam transcript that informed the paper is preserved at `_brainstorm_redteam.md` + `_brainstorm_round3_synthesis.md` for methodological auditability.