



Complex Knowledge Curation using Agentic Ontological Notebook Memory

A typed **knowledge graph** as agent memory — built, queried, and refined entirely in **natural language**.

Gully A. Burns · Paul Groth



SciKnow.io
Consulting

01 · MOTIVATION

Agents make ontological commitments. Most hide them.

Every LLM agent that persists memory *must* commit to a representation of what it knows.

In today's agent stacks those commitments are buried in prompts, tool definitions, and heuristics — implicit, untestable, hard to refine.

We make them explicit. Skillful Alhazen replaces flat vector or markdown memory with a typed knowledge graph in TypeDB. Domain concepts have names, types, and relations. The schema is the agent's vocabulary.

02 · THE NOTEBOOK SCHEMA

Two kinds of thing.

Following the Information Artifact Ontology (IAO), the schema separates real-world entities from representations of them. Every analysis is provenance-linked back to its source.

Domain things	
Companies · positions · diseases · genes · phenotypes · target systems · success criteria.	
INFORMATION CONTENT ENTITIES	
Artifact	Raw captured content — API responses, HTML, PDFs, repos.
Fragment	Structured extraction — a requirement, phenotype association, feature-evidence quote.
Note	Agent analysis — fit scores, mechanism hypotheses, comparative syntheses. Notes can link to notes.

03 · SYSTEM ARCHITECTURE

Three layers, one schema.

Skills are composable: each bundles a TypeDB schema extension, Python scripts for external APIs, and two-tier instructions (a slim **SKILL.md** for selection, full **USAGE.md** for execution).

1. Agent

Claude Code (interactive) or OpenClaw (always-on). Natural-language only — the user never writes a query.
- ▼
2. Skills

Composable modules — schema.tql · scripts · SKILL.md · USAGE.md · dashboard.
- ▼
3. TypeDB Ontological Memory

Type hierarchies · role-based relations · schema-level logic. Closed-world, declarative, queryable.

04 · THREE DEMONSTRATIONS

One schema, three working domains.

Each skill ingests artifacts, extracts fragments, and writes agent notes — against the *same* ontological backbone.

SKILL A

Job Hunt

Personal career tracking.

Ingest a job posting URL → artifact. Extract requirements → fragments. Generate a fit-analysis note against a user-maintained skill profile. Over time, accumulate a queryable landscape of the job market.

ASK CLAUDE
"What skill gaps appear across my top 5 prospects?"

SKILL B

Tech Recon

Goal-driven technology investigation.

Success criteria are *first-class entities*. Candidate systems flow through a typed pipeline — **candidate** → **confirmed** → **ingested** → **analyzed** — so investigation progress is itself a queryable graph property.

ASK CLAUDE
"Map agentic memory systems against my four success criteria — how do MemGPT, Mem0, Zep, Letta compare?"

SKILL C

DisMech

Rare-disease mechanism curation.

Mirrors the Monarch Initiative's DisMech LinkML schema into TypeDB. **797 disorders** with phenotypes (HPO), causal genes, treatments (MAXO), and PubMed evidence — now a single typed graph rather than YAML files scanned per query.

ASK CLAUDE
"Which diseases involve WNT/β-catenin mechanisms but have no documented treatment?"

05 · RESULT

When the question is structural, RAG cannot reach it.

DisMech benchmark — 13 questions, 3 categories, same corpus. Ground truth computed deterministically by scanning YAML (no LLM in scoring).

Pathway aggregation

"How many diseases involve WNT/β-catenin mechanisms?"		
TypeDB	<div><div></div></div>	0.75
RAG	<div><div></div></div>	0.00

Absence detection

"Which diseases have phenotypes but no genetic entries?"		
TypeDB	<div><div></div></div>	0.68
RAG	<div><div></div></div>	0.00

Global ranking

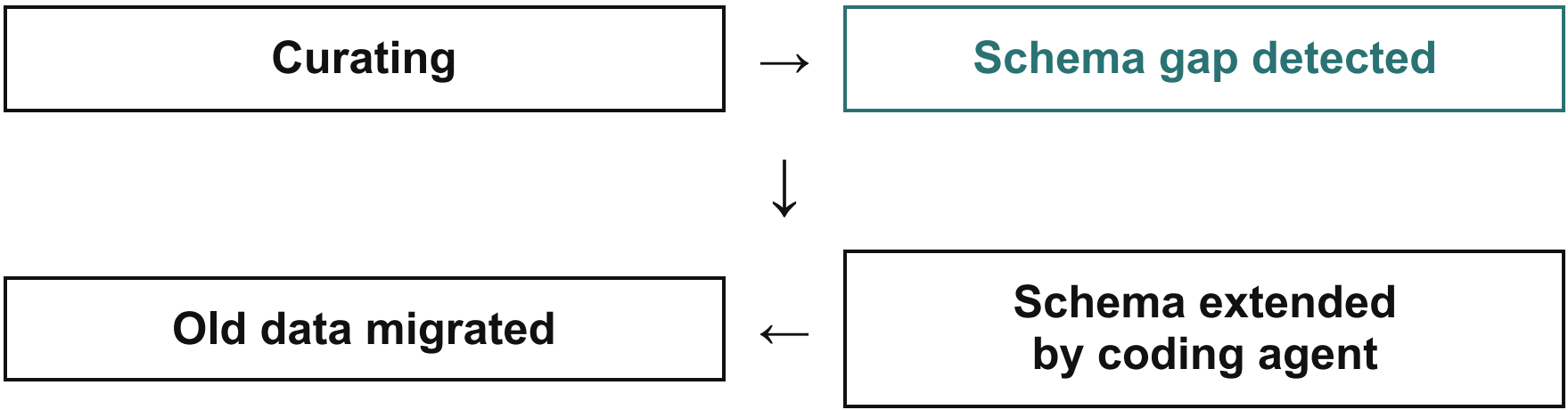
"Top 5 diseases by number of mechanisms?"		
TypeDB	<div><div></div></div>	0.77
RAG	<div><div></div></div>	0.03

RAG's failures are **architectural** — aggregation requires counting; absence requires evidence of non-existence; ranking requires ordering by structural property. Typed graphs have these primitives natively.

06 · SCHEMA EVOLUTION

The graph grows from use.

When the agent encounters something the current schema cannot represent, it records the gap as a note *and* files a GitHub issue. The coding agent then extends the schema; GLAV mapping rules migrate prior data; curation continues.



07 · TRY IT AT THE BOOTH

Scan, ask, watch the graph grow.

- "Ingest this URL as a job posting and show me the fit analysis."
- "Show the provenance chain from this note back to its source."
- "Switch to tech-recon and map a topic of your choice."

