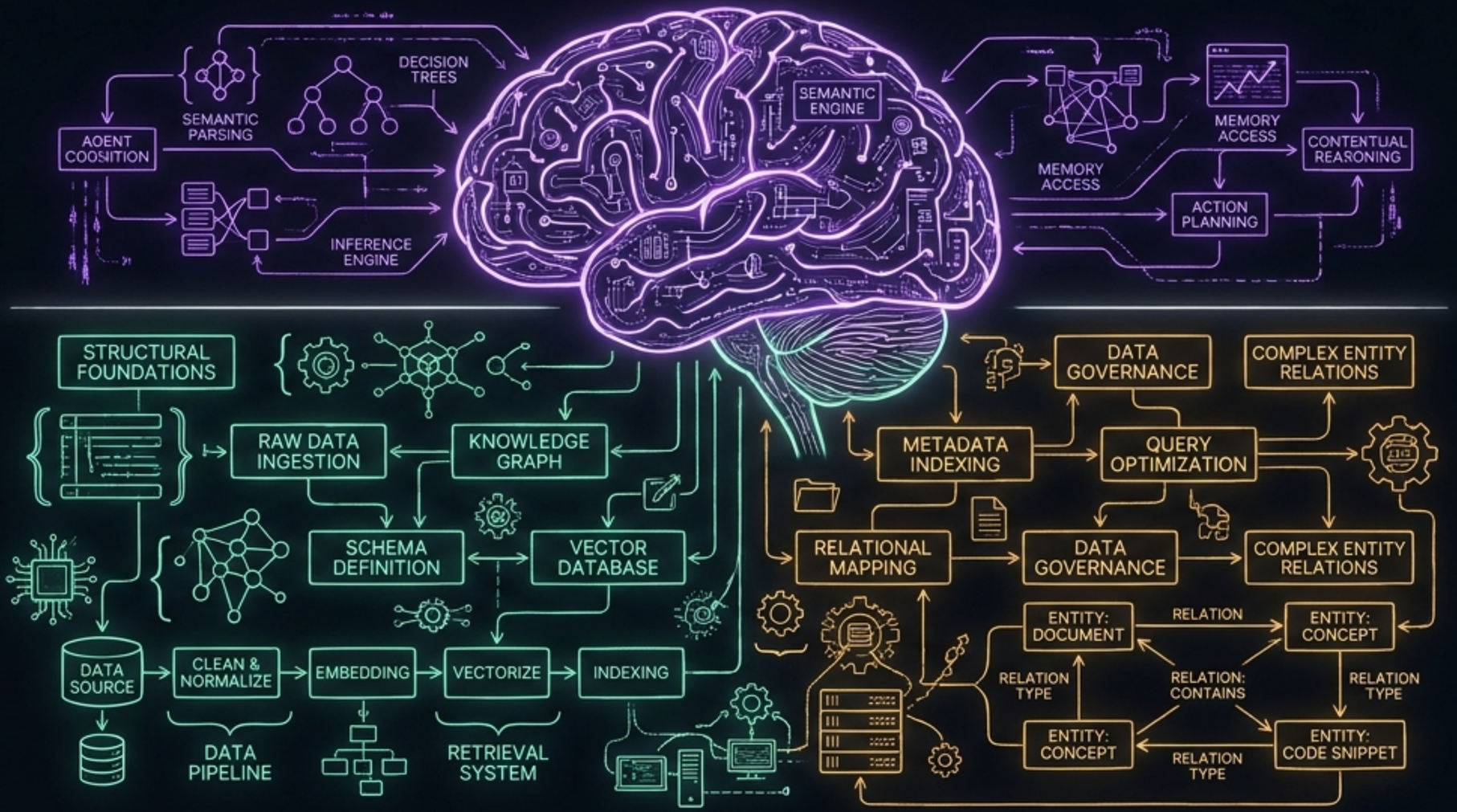
Knowledge Engineering for AI Agents

The definitive guide to building enterprise agent tools and retrieval pipelines.

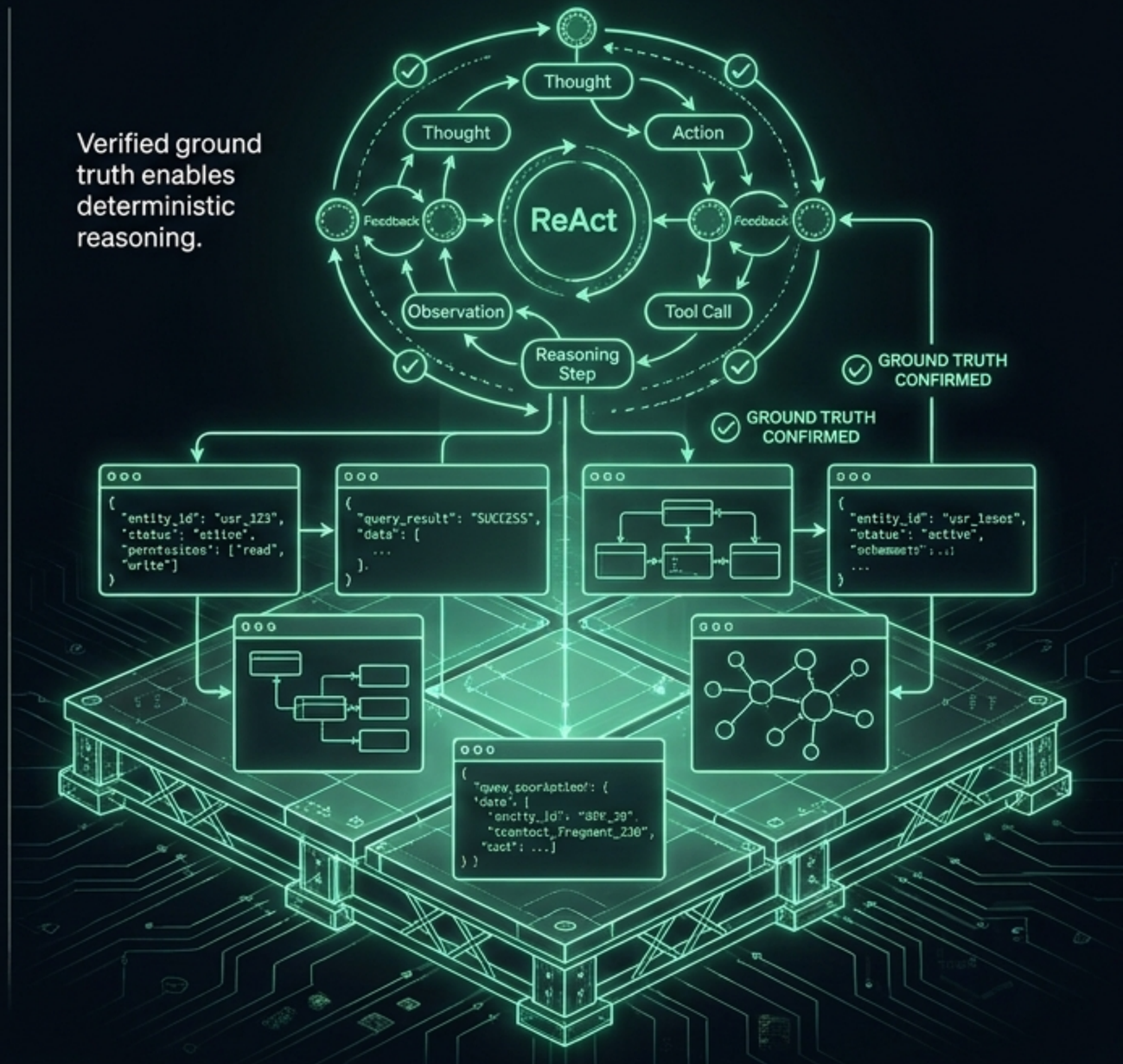


An agent is only as reliable as the ground truth returned by its tools.

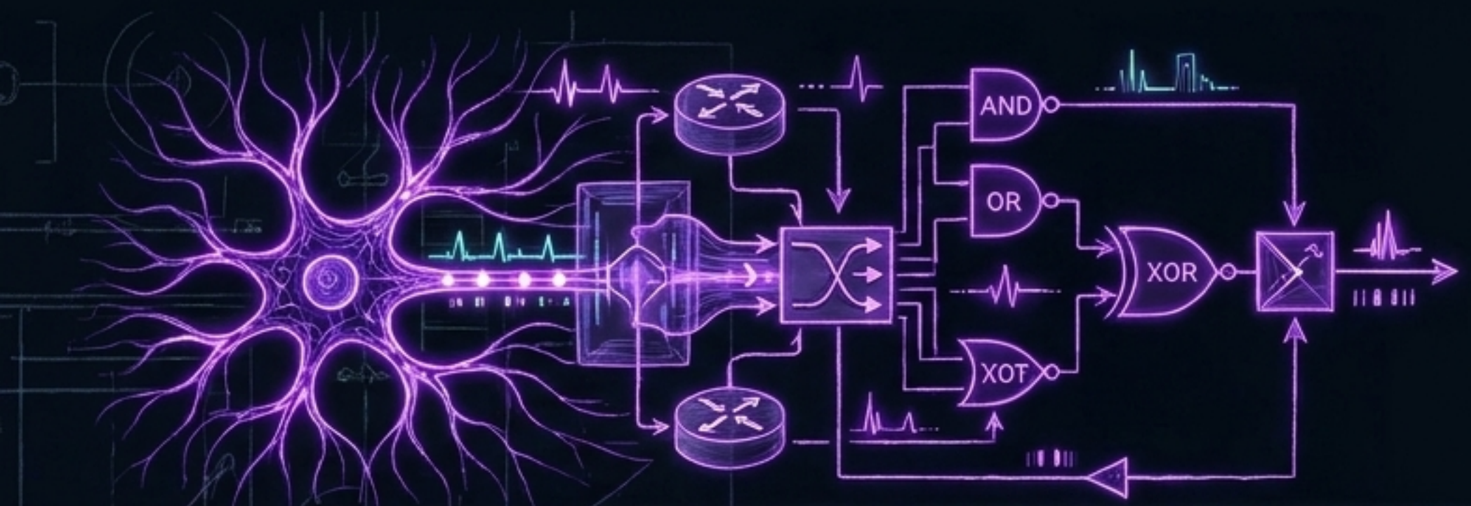


Even the most sophisticated reasoning loop (ReAct, Orchestrator-Workers) will confidently hallucinate if the knowledge layer layer drops tables or severs clauses.

Verified ground truth enables deterministic reasoning.

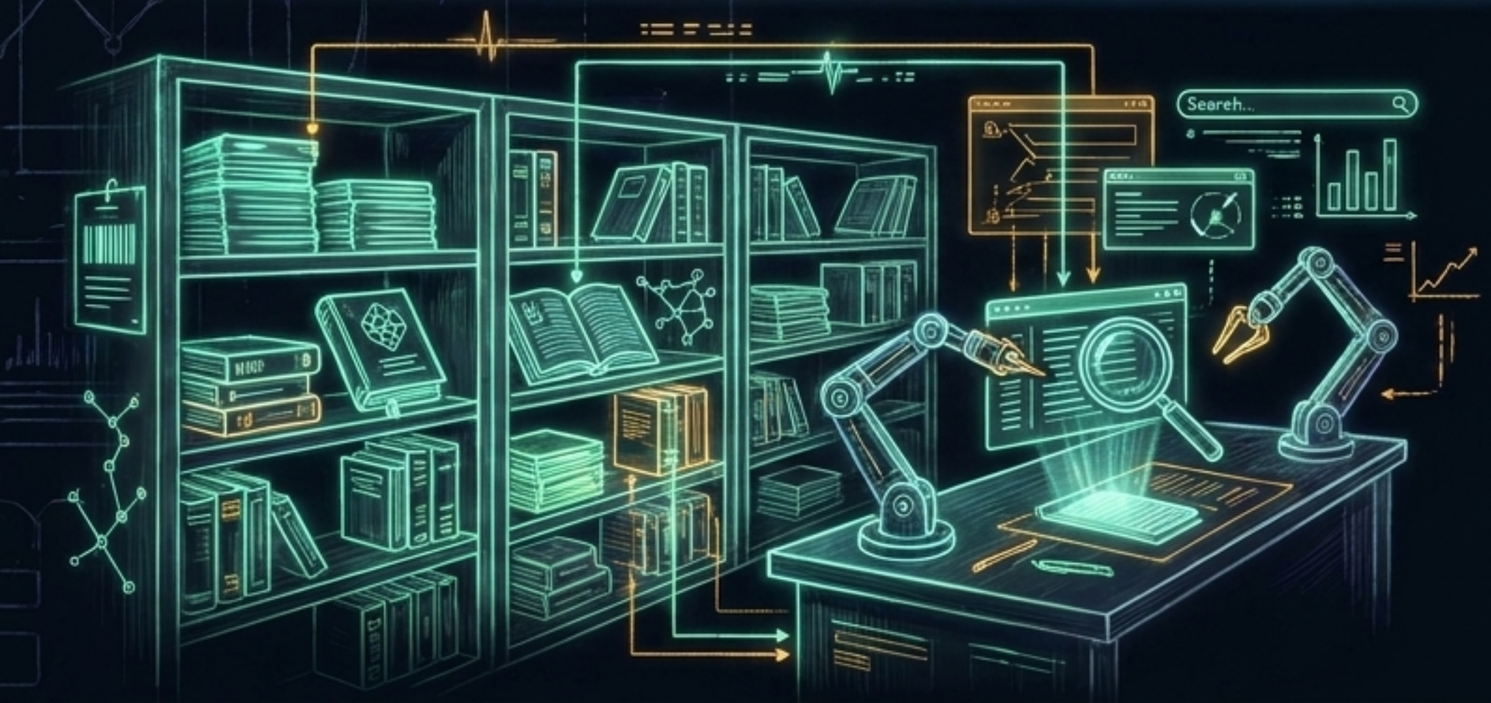


The Mental Model: Separating the Nervous System from the Library



Agent Design Patterns (ADP)

Builds the NERVOUS SYSTEM — how the agent thinks, loops, and asks for human approval.



Knowledge Engineering

Builds the **LIBRARY AND CATALOG** — how raw files become clean books, , indexed by topic, keyword, and citation network, exposed via specialized tools.

The Prototype vs. Production Chasm

Dimension	Naive RAG (Prototype)	Knowledge Engineering (Production)	Memory Hook
Parsing	Raw text stream (drops tables & layout)	Layout-aware vision/OCR (Docling, Document AI)	Read the page layout, not just the raw bytes
Chunking	Fixed size (e.g., 500 tokens)	Structure-aware (split by section/clause)	Cut at chapter seams, never mid-sentence
Context	Raw text only	Rich metadata & provenance	Every quote carries its birth certificate
Search	Vector similarity only	Hybrid (Vector + BM25) + Filters + Graph	Meaning + exact IDs + relationship web
Agent Access	Passive prompt stuffing	Strongly-typed Retrieval Tools (MCP)	Specialized scalpel tools, not one blunt shovel

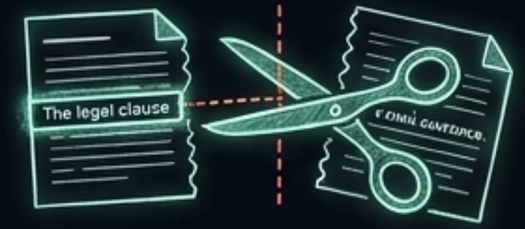
The Five Structural Gaps in Enterprise Knowledge



1. Silent parsing corruption: Standard extractors strip table grids and scramble multi-column layouts.



2. Severed logical context: Fixed-token chunking slices legal clauses away from exceptions.



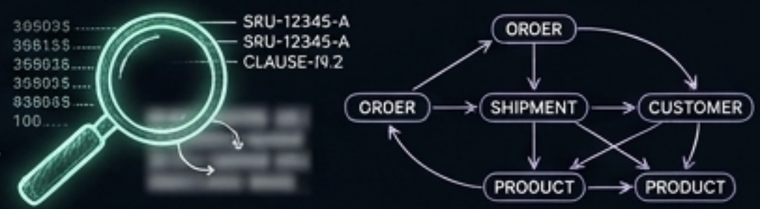
3. Permission & version blindness: Pure vector similarity ignores expired policies or tenant ACLs.

3. Permission & version blindness: Pure vector similarity ignores expired policies or tenant ACLs.

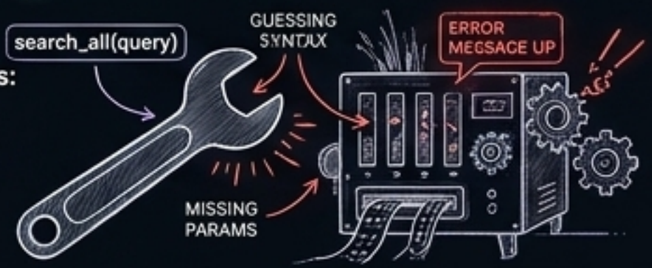


4. Lexical & relational blindness: Dense embeddings miss exact SKUs, clause IDs, and multi-hop dependencies.

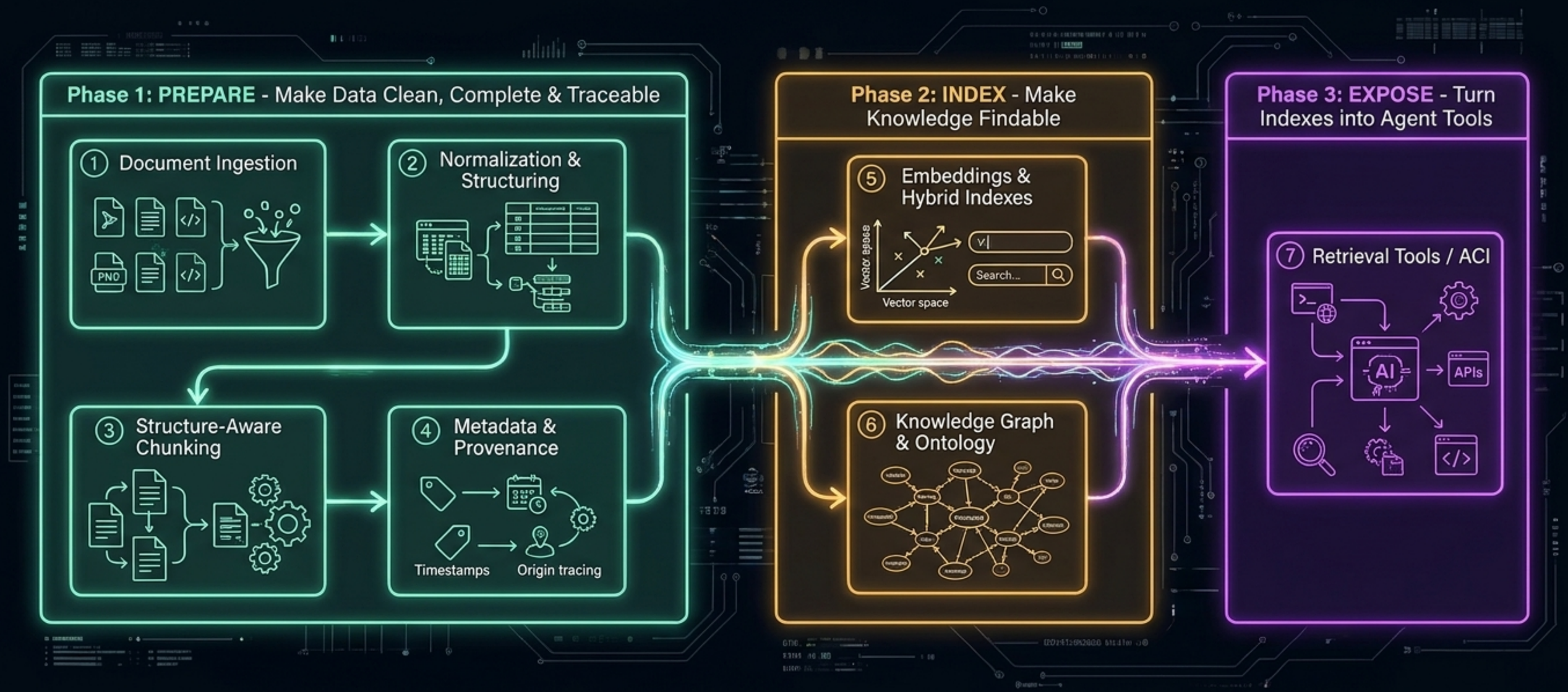
4. Lexical & relational biindness: Dense embeddings miss exact SKUs, clause IDs, and multi-



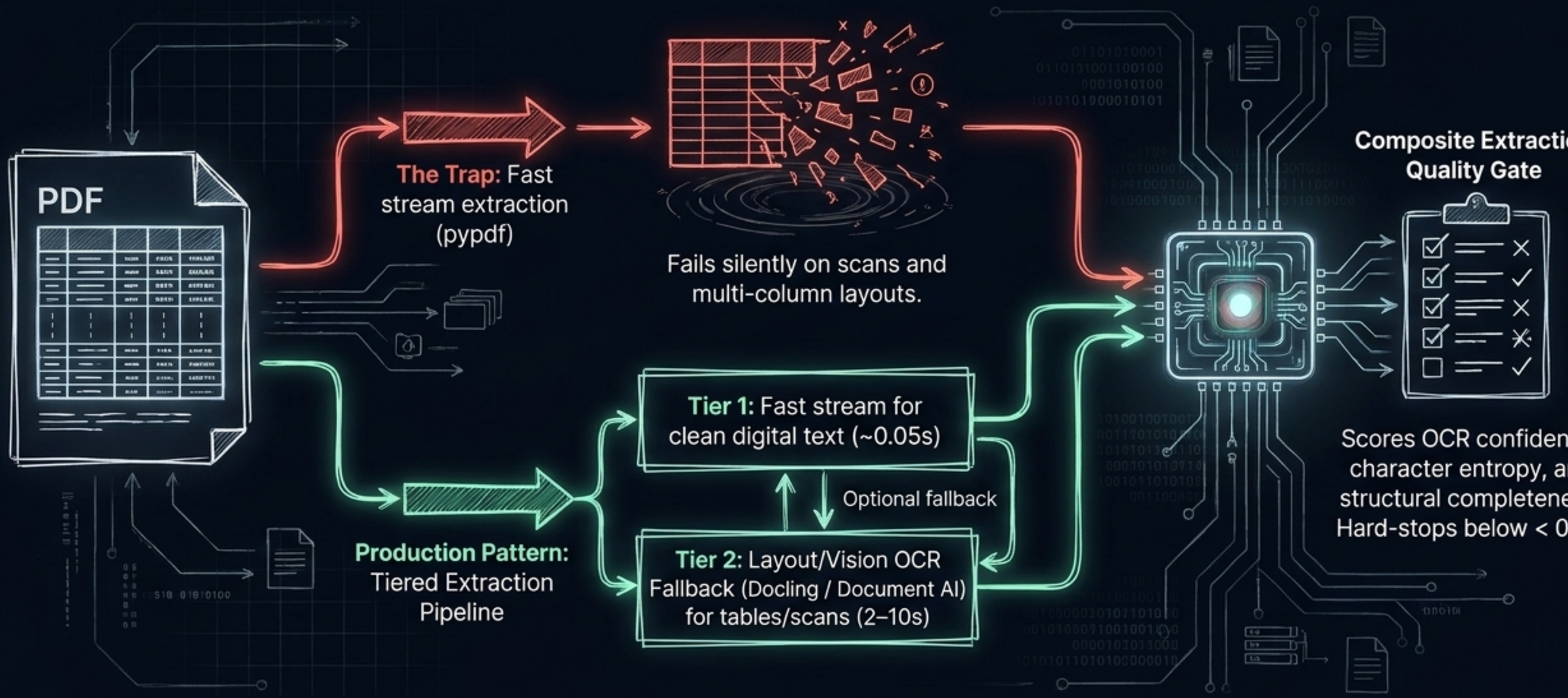
5. Ambiguous tool interfaces: A generic search_all(query) tool forces models to guess syntax.



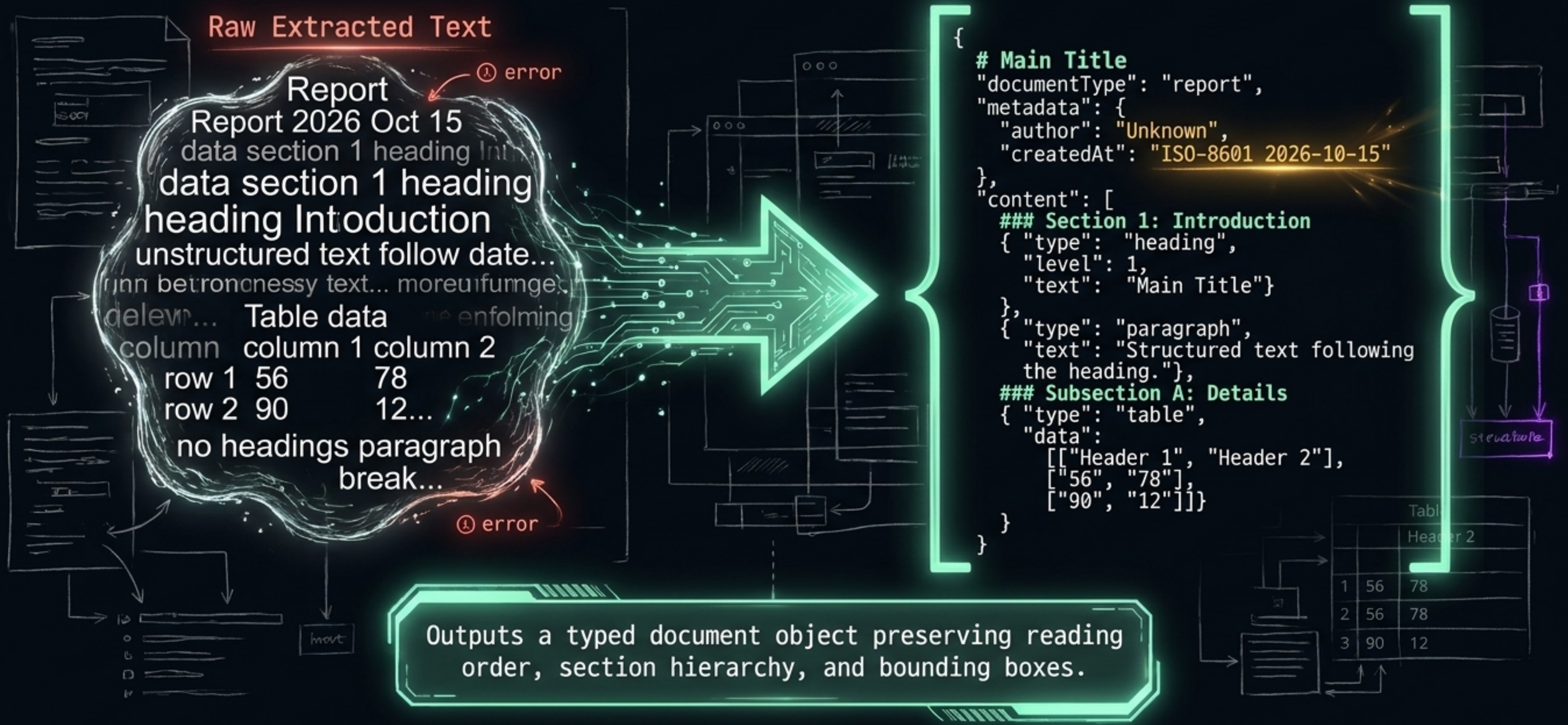
The Architecture Blueprint: A 3-Phase, 7-Step Framework



Step 1: Ingestion Pipeline & Epistemic Gating

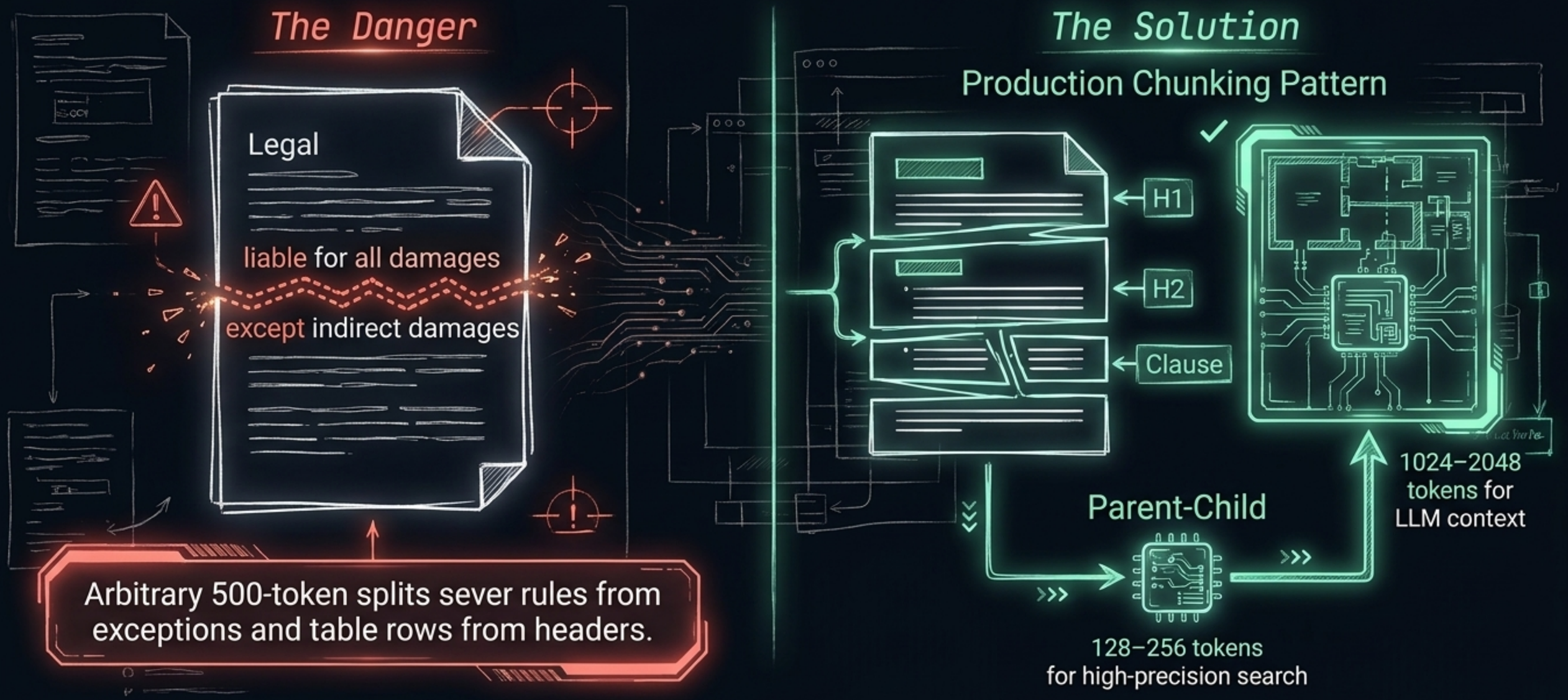


Step 2: Normalization and Structural Hierarchy

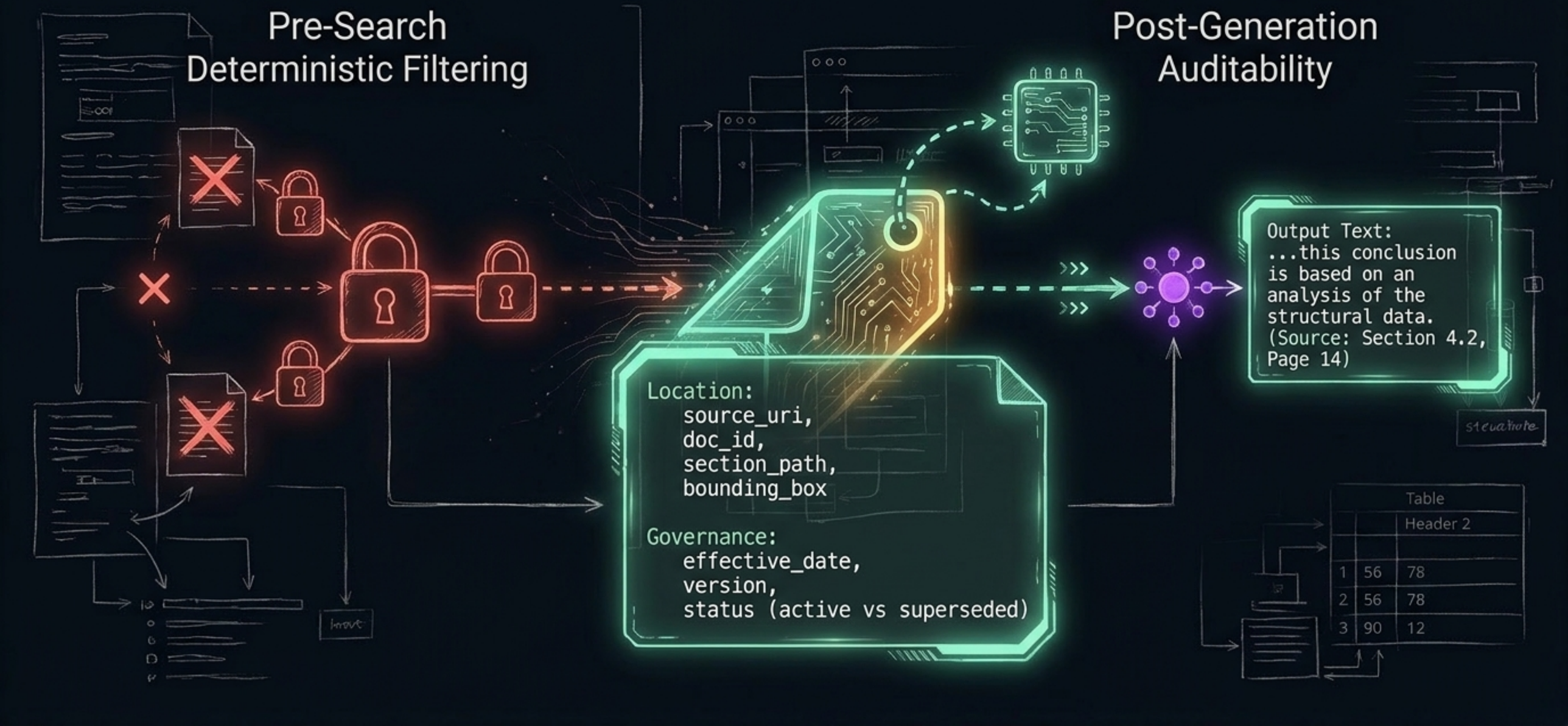


Outputs a typed document object preserving reading order, section hierarchy, and bounding boxes.

Step 3: Structure-Aware Chunking



Step 4: Metadata and Provenance Envelopes



Phase 2: Indexing for Meaning, Keyword, and Relationship

Meaning



Dense Vector Search (768–3072 dims)

Captures semantic intent and synonyms (e.g., “termination for cause” = “ending agreement”).

Keyword



Sparse Lexical Search (BM25)

Captures exact alphanumeric strings, SKUs, error codes, and proper nouns.

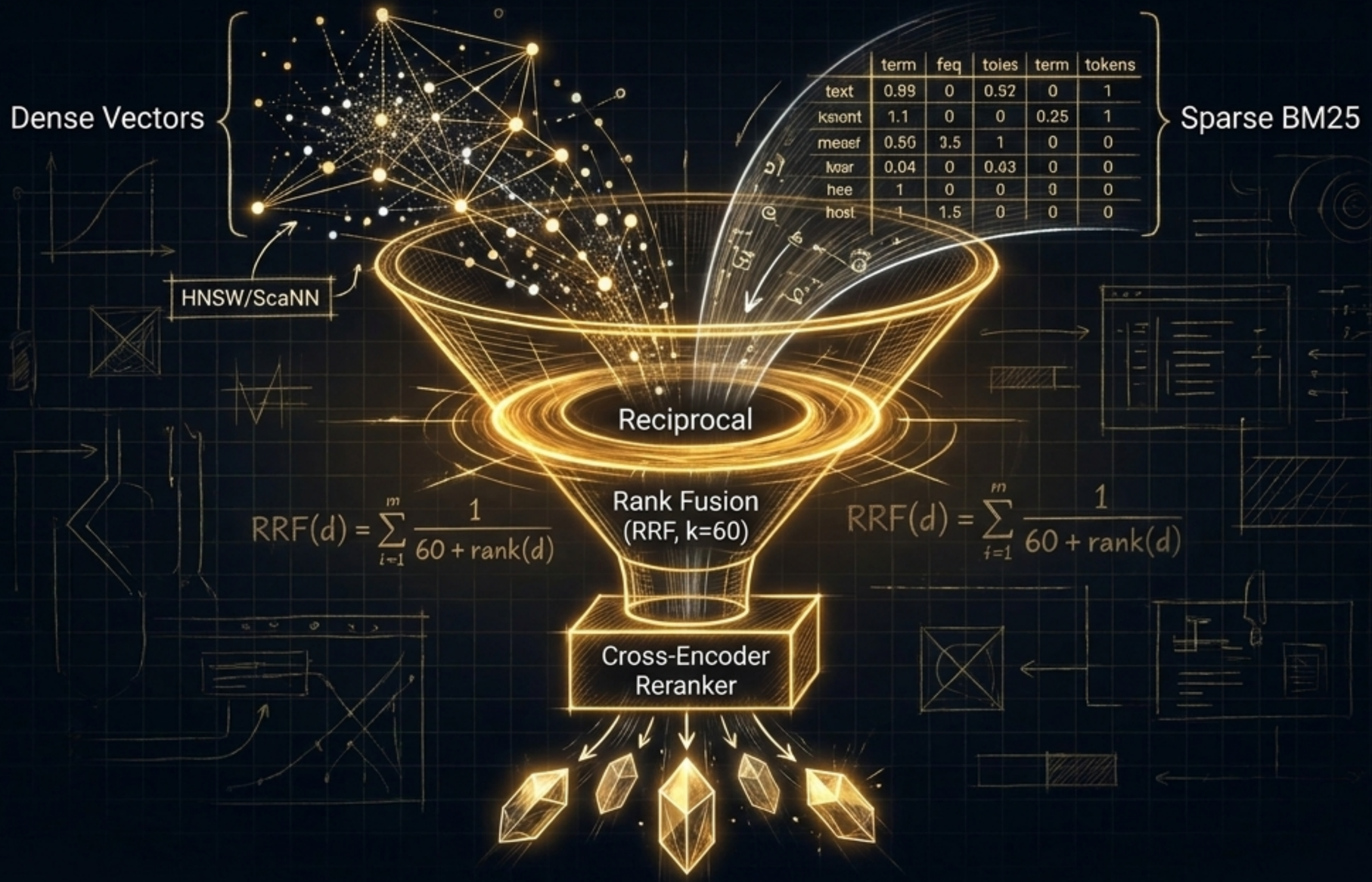
Relationship



Knowledge Graph

Multi-hop relational reasoning across documents.

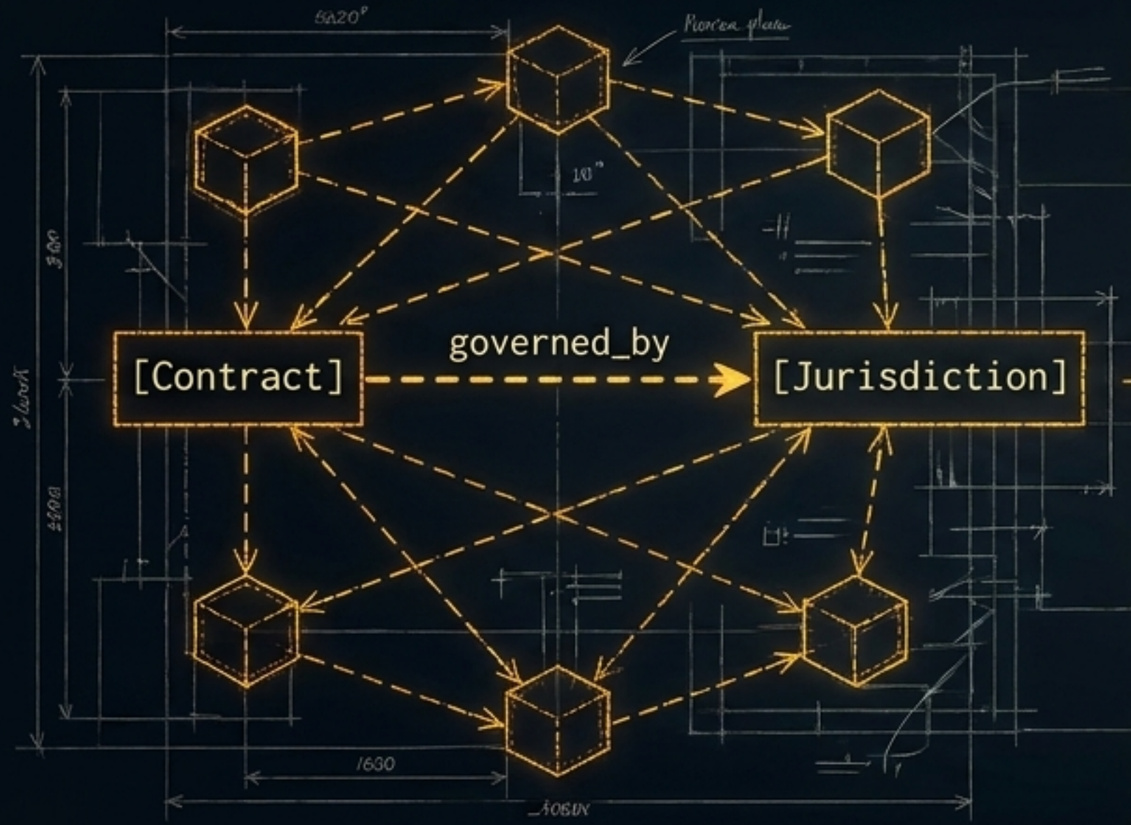
Step 5: Hybrid Search Pipelines



Top-N highest-precision passages.

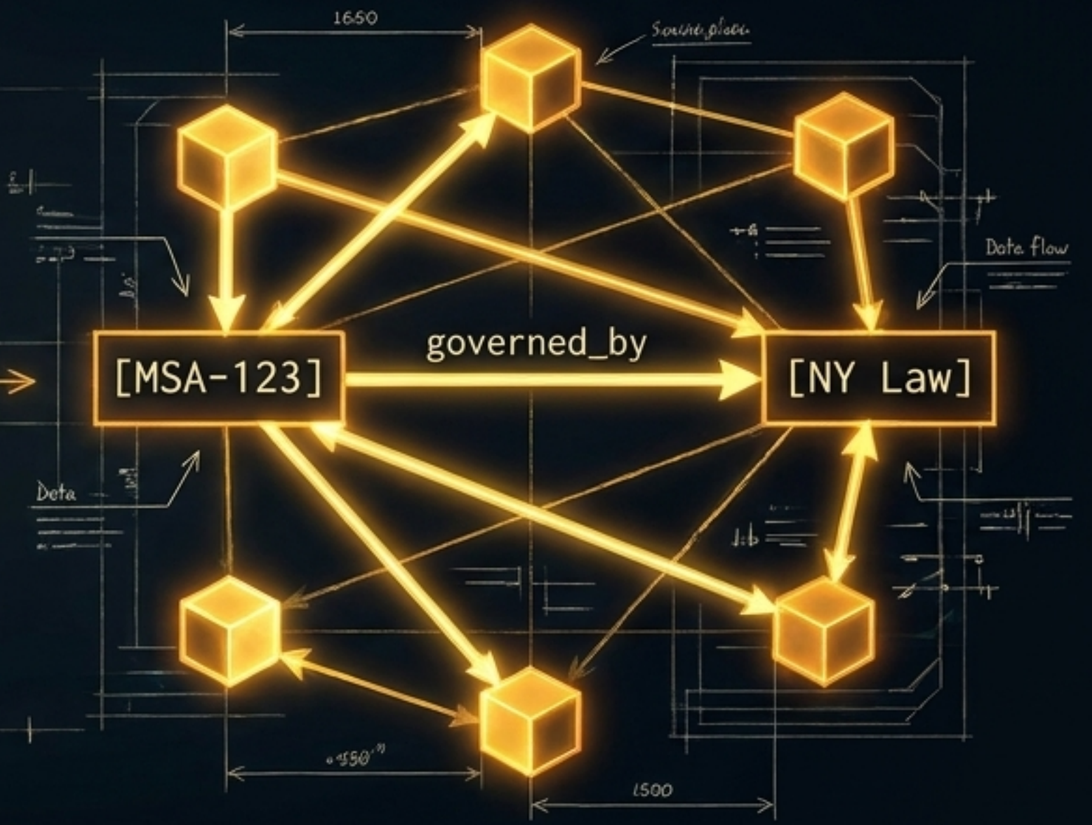
Step 6: Ontology vs. Knowledge Graph

Ontology (The Schema / TBox)



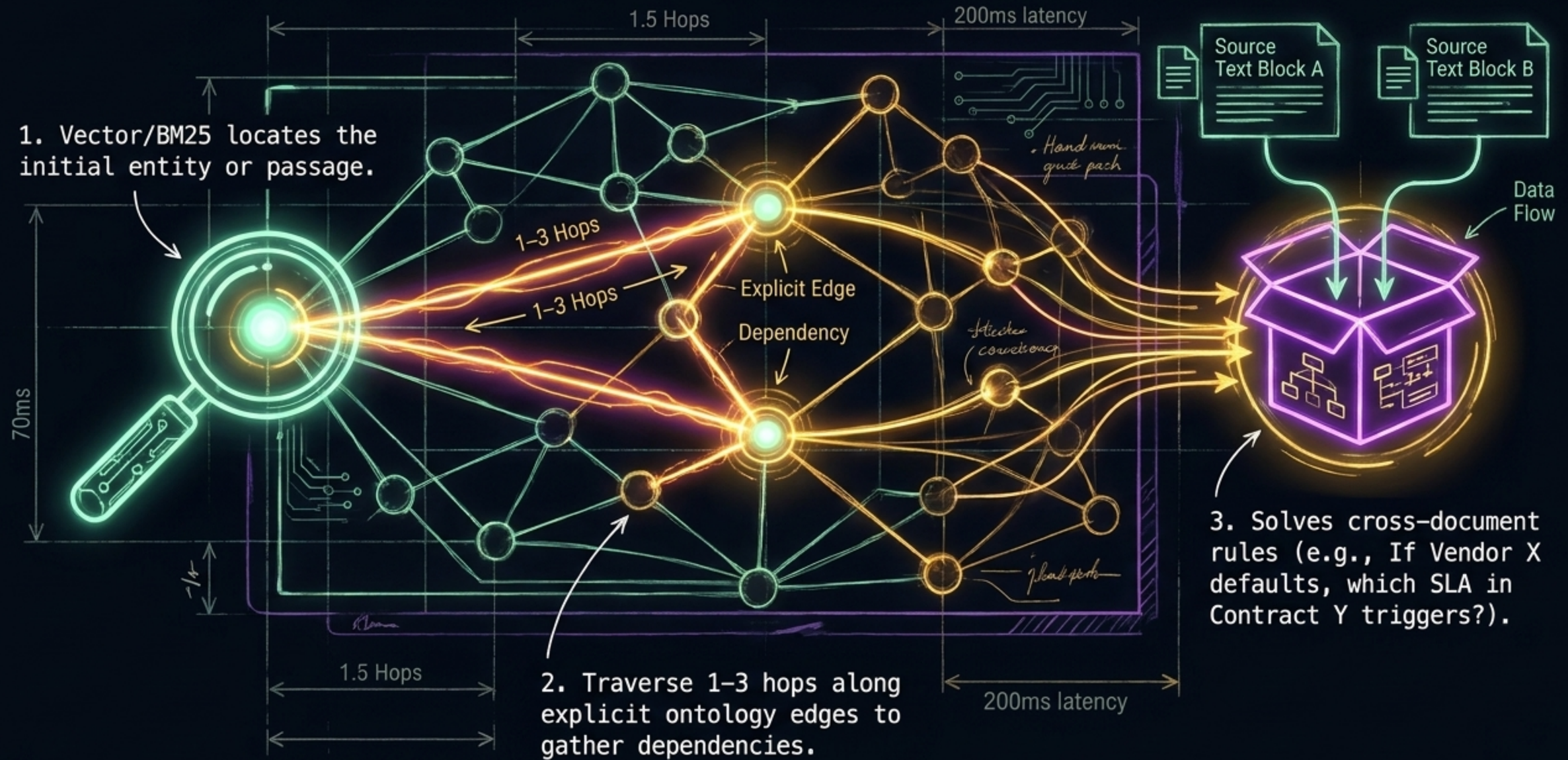
Defines allowed classes and predicates (RDFS, OWL).

Knowledge Graph (The Instances / ABox)



Stored in Neo4j/Spanner Graph, queried via Cypher.

The Hybrid GraphRAG Traversal Pattern



Phase 3: The Agent-Computer Interface (ACI)

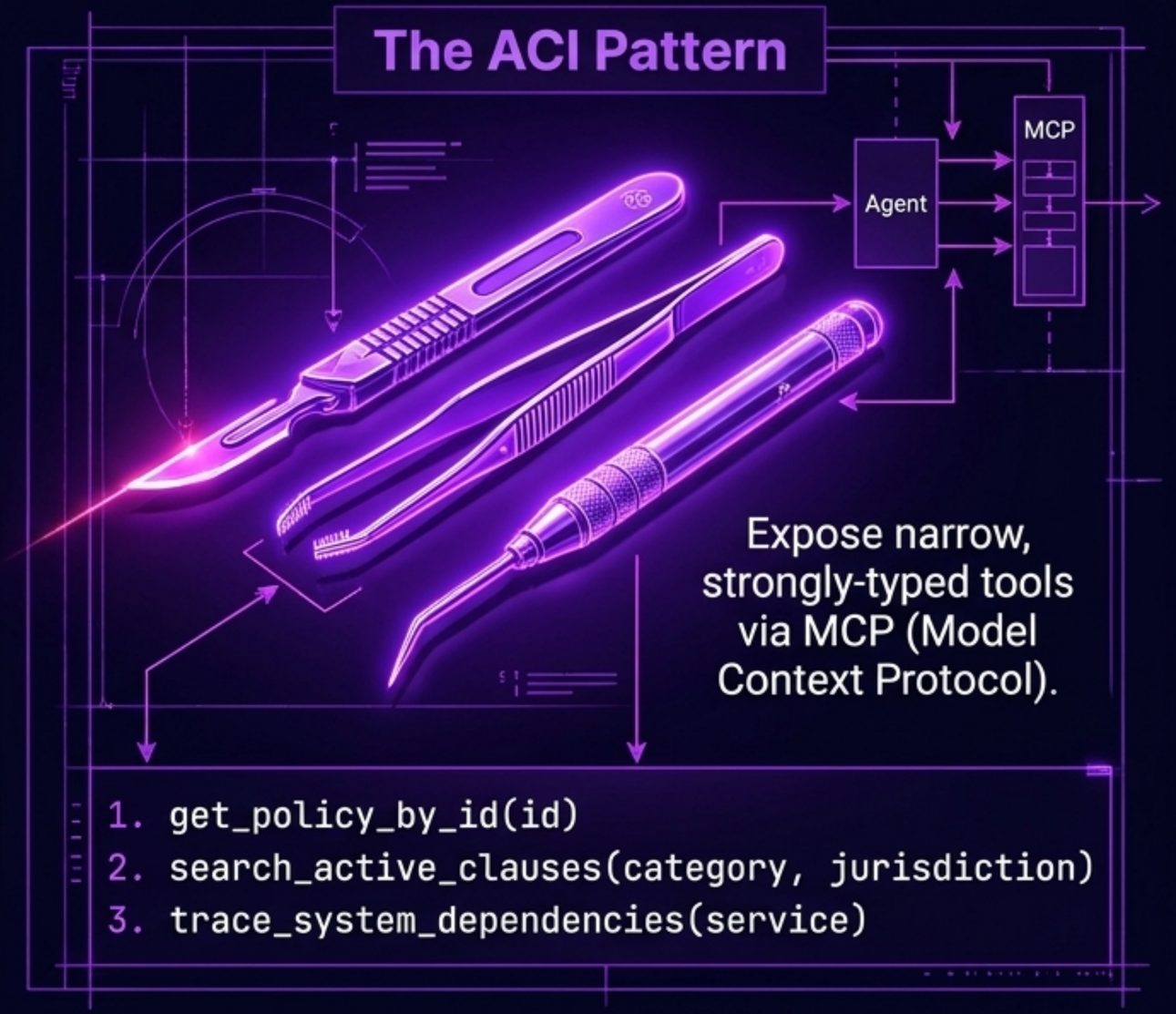
The Anti-Pattern



```
search_all(query: str)
```

Forces model to guess syntax, ignores deterministic constraints, floods context.

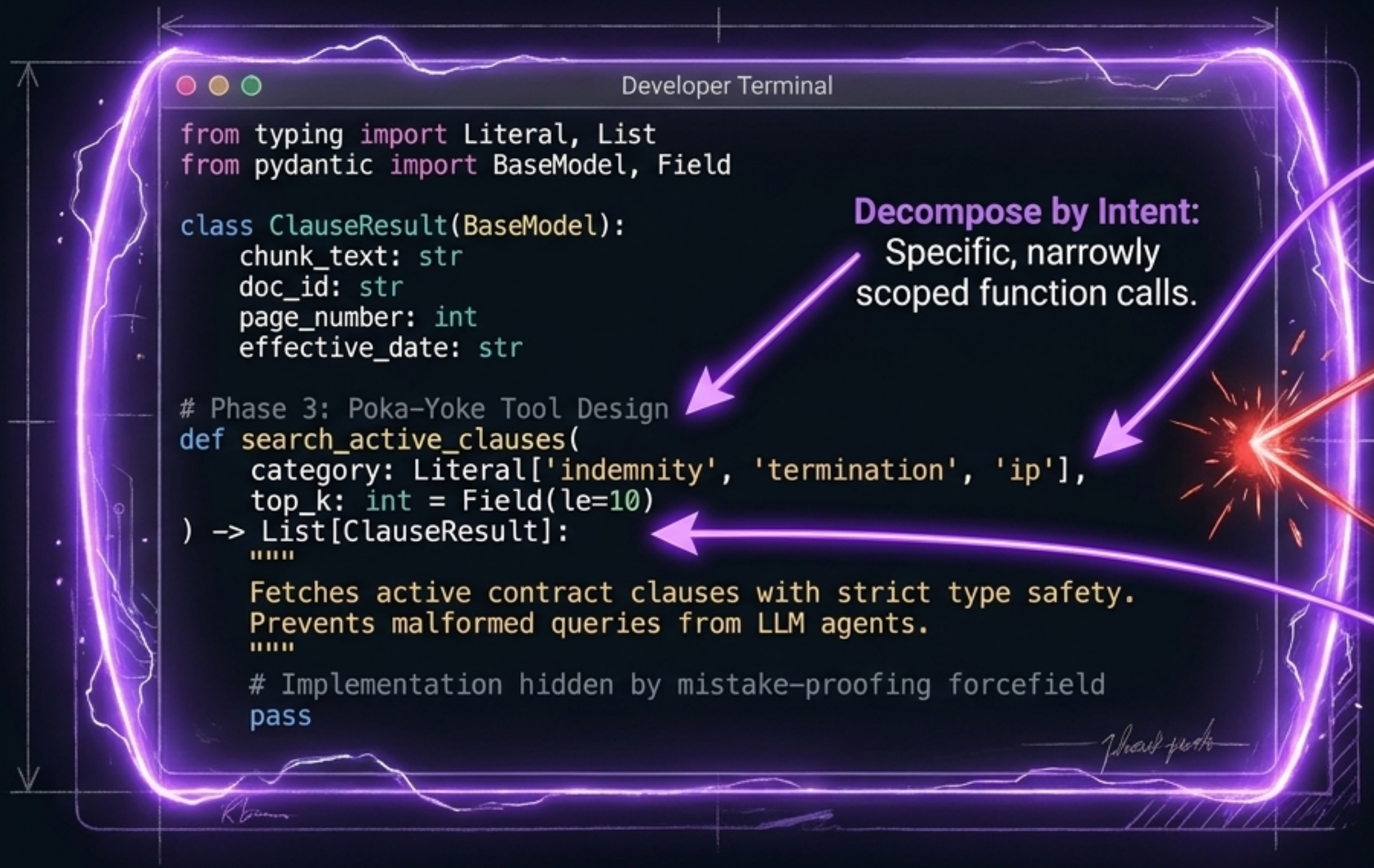
The ACI Pattern



Expose narrow, strongly-typed tools via MCP (Model Context Protocol).

```
1. get_policy_by_id(id)
2. search_active_clauses(category, jurisdiction)
3. trace_system_dependencies(service)
```

Step 7: Poka-Yoke Tool Design Rules



Agents must receive structured data to verify citations deterministically.

Structured Data Tools: Parameterized SQL vs. NL2SQL



Parameterized Query Tool

Example: get_refunds(region)

Who writes SQL: Engineer



Use when: Known, recurring customer paths.

Controls: Bound parameters, typed arguments.

NL2SQL over Curated Layer

Who writes SQL: Model, against approved views.

Use when: Analyst domain exploration.

Controls: Metric glossary, verified examples.

Open NL2SQL

Example: execute_sql

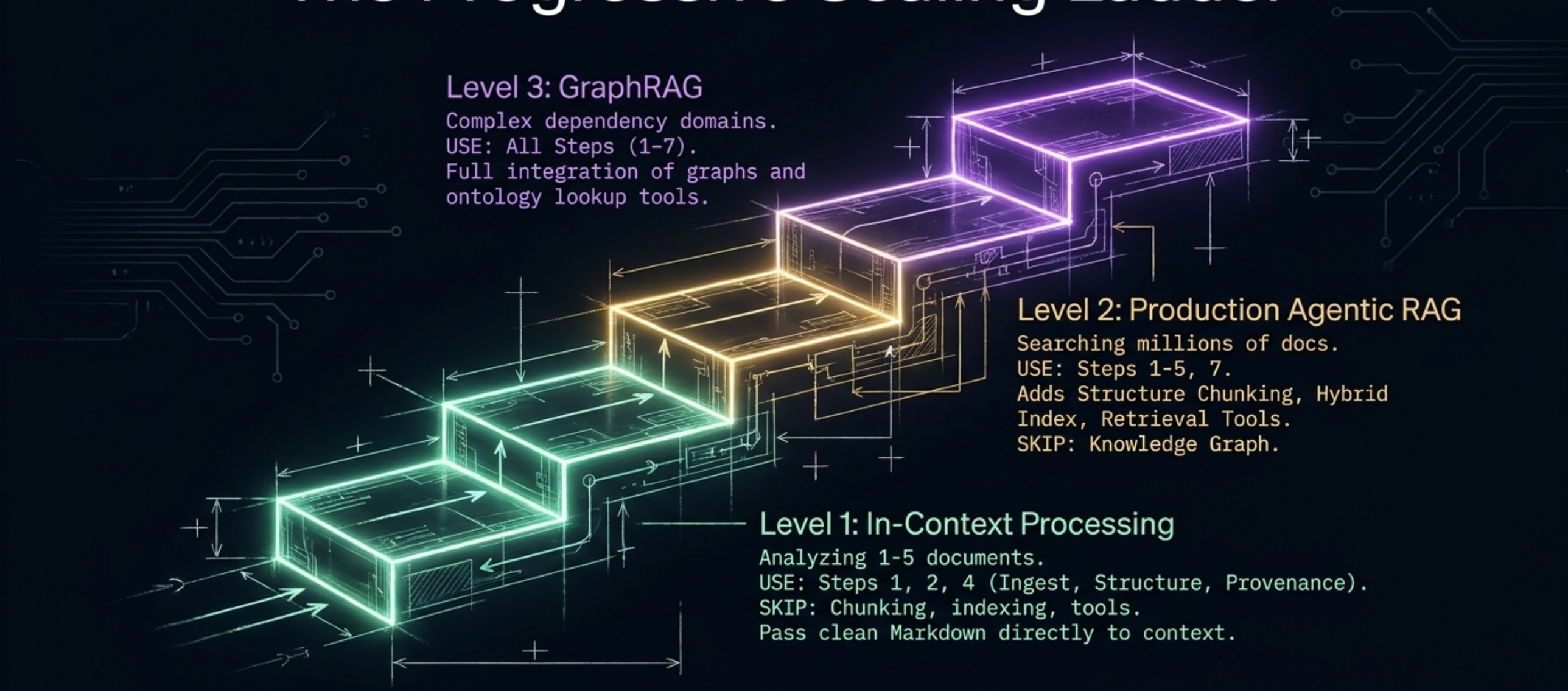
Who writes SQL: Model, any table.

Use when: Trusted internal sandbox only.

Controls: Human review required.

HUMAN REVIEW

The Progressive Scaling Ladder



Level 3: GraphRAG

Complex dependency domains.
USE: All Steps (1-7).
Full integration of graphs and ontology lookup tools.

Level 2: Production Agentic RAG

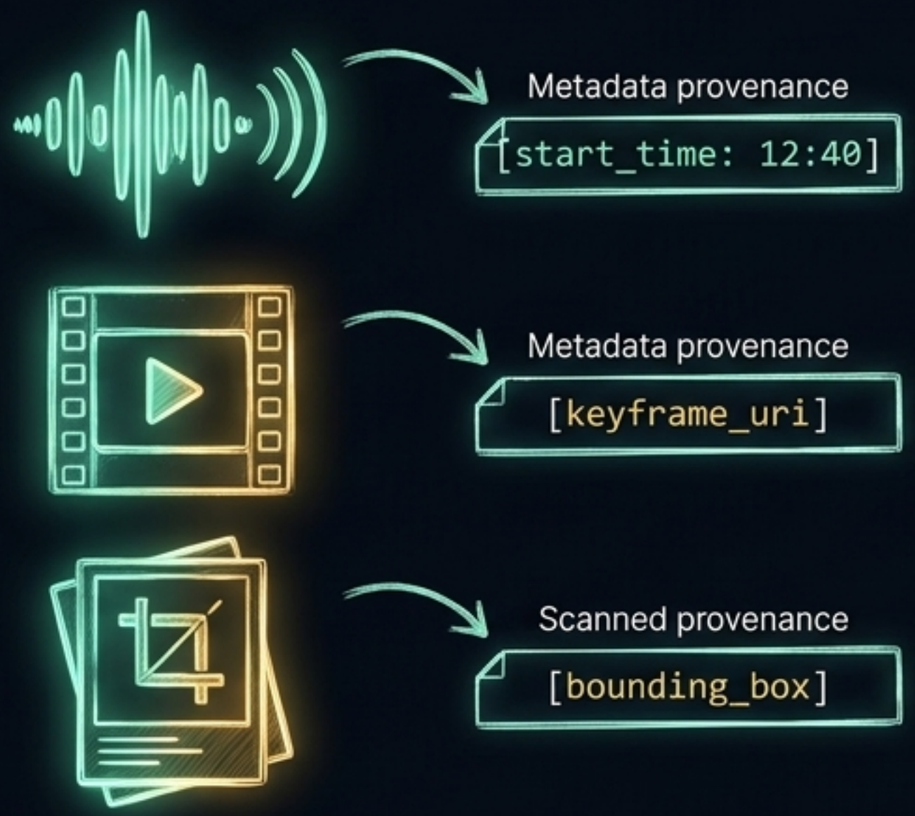
Searching millions of docs.
USE: Steps 1-5, 7.
Adds Structure Chunking, Hybrid Index, Retrieval Tools.
SKIP: Knowledge Graph.

Level 1: In-Context Processing

Analyzing 1-5 documents.
USE: Steps 1, 2, 4 (Ingest, Structure, Provenance).
SKIP: Chunking, indexing, tools.
Pass clean Markdown directly to context.

Operating the Pipeline: Multimodality & Security Gates

Multimodal Transformation



Rule: Turn every media file into time- or region-addressable units.

Identity-Aware Access



Enforce ACLs in code, not via prompts. Redact before indexing.

Evaluation & The Knowledge Boundary

Evaluating the Golden Set



Retrieval
(recall@k, MRR)
- Measuring finding the right chunk



Golden Set



Generation
(groundedness, citation accuracy)
- Measuring not hallucinating

The Library

Massive, shared across users, ACL-filtered, authoritative, versioned ground truth





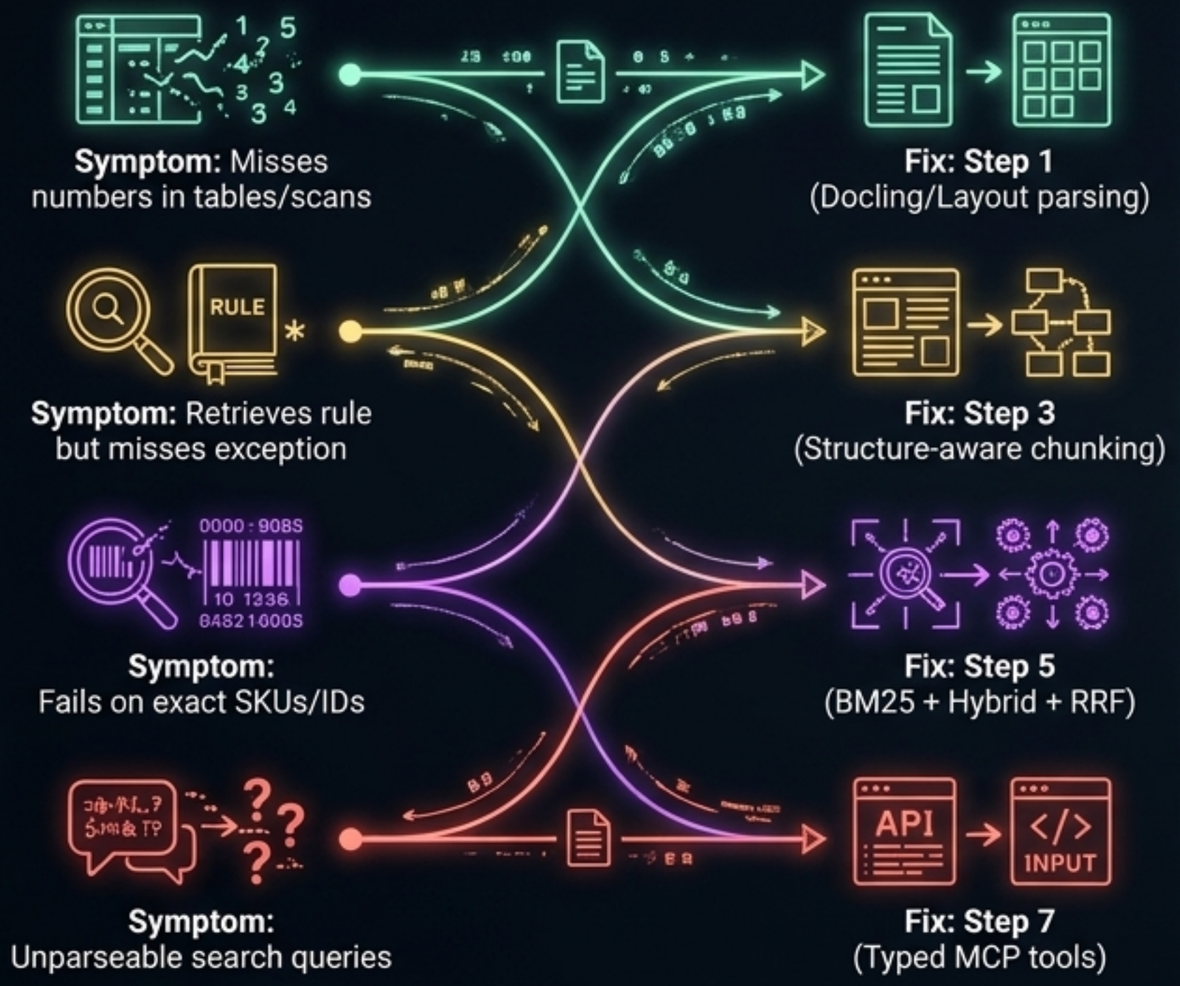
The Notepad

Small, per-session, isolated, user-specific extracted facts with TTL

Never write unverified conversation facts into the shared knowledge base

Diagnostic Cheat Sheet & Pocket Reference

Diagnostic Troubleshooting Tree



The Framework Summary

