DISCUSSION FRAMEWORK FOR FOUNDERS & BUILDERS

Context Graphs: Open Problems Worth Solving

The diagnosis is right — enterprises are drowning in data and starving for decision context. But the path from diagnosis to working systems runs through hard, unsolved problems.

CG1 & CG2

Foundation Capital — "AI's trillion-dollar opportunity" (Dec 2025) & "The compounding asset" (Apr 2026)

Critique

Sid Shah — "I Want Context Graphs to Work" (Apr 2026)

SLIDE 1 OF 2 — HISTORICAL CONTEXT

Three Waves of Graphs — And Why They Failed

The idea that structured knowledge will transform enterprises has been tried three times.

	Wave 1: Semantic Web 2001–2008 · RDF, OWL, Ontologies	**Wave 2: Enterprise KGs** 2012–2022 · Neo4j, Property Graphs	**Wave 3: LLM-Powered CGs** 2024–now · LLM extraction, Agents
Creation	HARD Manual ontology curation by SMEs. Months to build a single enterprise ontology. RDF couldn't express most real-world constraints.	MEDIUM Property graphs more pragmatic but schema design still required specialist data modelers. Better than Wave 1, still not accessible.	EASY LLMs extract entities & relationships from unstructured text in hours. No manual ontology design. Point at a corpus, get a graph.
Maintenance	IMPOSSIBLE Business changes outpaced manual updates. No one budgeted for continuous ontology upkeep. Small errors cascaded.	COSTLY Schema evolution manual & tedious. Data drift happened silently. Graph created once, maintained never. No SLA for freshness.	UNCERTAIN Agents may self-update within their workflows. But most context is still created outside agent frameworks — meetings, calls, Slack. Unproven at scale.
Governance	UNOWNED IT said business problem. Business said data problem. Data said governance problem. Nobody owned it. Technology worked; org model didn't.	STILL UNOWNED Same vacuum. Most orgs couldn't move past pilots. Production deployments concentrated in regulated domains with forced compliance.	HARD Most enterprise context lives in human-human interactions that aren't documented. Zoom transcripts capture words, not reasoning. Works for narrow domains, unclear at scale.
Outcome	Succeeded in pharma, bioinformatics. Failed as general enterprise approach.	Production in fraud, finance, pharma. Disproportionately in regulated/constrained domains.	Too early to tell. Creation finally solved. The other two problems remain open.

SLIDE 2 OF 2 — THE PATTERN

The Same Pattern Every Time

Each wave solved creation a bit better. None solved the other two.

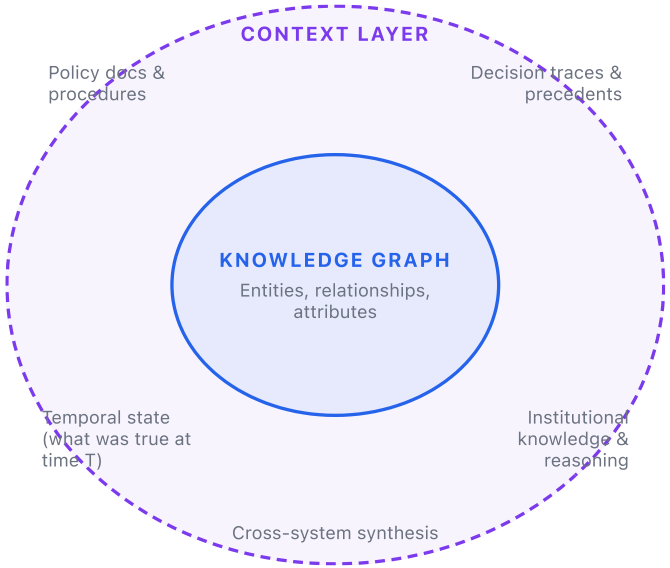
Creation	Manual / SMEs	→	Schema modeling	→	LLM extraction	✓ Solved
Maintenance	Impossible	→	Costly / silent drift	→	Agents help, but...	? Unsolved
Governance	Unowned	→	Still unowned	→	Fundamentally human	✕ Unsolved

Wave 3's bet: Agents in the execution path will generate context as a byproduct of doing work. But we're in a multi-year hybrid equilibrium where humans and agents coexist. The governance gap — who owns context, who keeps it true — has never been solved by any wave. **That's the problem space we're here to discuss.**

DEFINITIONS

What Exactly Is a "Context Graph"?

The current discourse conflates two things. We need to be precise.



Foundation Capital's definition: Decision traces

A record of what inputs were gathered, what policy was evaluated, who approved, and what state was written. Their deal desk example: agent proposes discount, gathers precedent, routes exception, records outcome.

The problem: This is narrow. How many workflows actually produce clean decision traces? Deal desks, SOX audits, code reviews, underwriting. Not many others.

A broader (more useful) definition

Context Graph = KG + everything that gives meaning to those entities: policy documents, sales motion procedures, quote-to-cash workflows, temporal state, institutional knowledge. Decision traces are one *type* of context, not the whole thing.

Debate question for the group:
Is "context graph" just "knowledge graph done right with provenance and time"? Or is it architecturally different — requiring a new data structure (traces + reasoning + time)?

PRACTICAL EXAMPLE — PART 1

What Context Does an Agent Actually Need?

Example: An AI agent that recommends which opportunities a sales rep should prioritize this week.

1 — Entity Data (KG)
Open opportunities in the rep's patch, accounts, contacts, contract history, product SKUs. Lives in CRM today.

2 — Internal Signals
Product usage telemetry, churn risk flags, seller notes, support tickets (MTTR, Sev1s), exec attendance at QBRs.

3 — Third-Party Signals
Competitor mentions in earnings calls, digital transformation announcements, hiring patterns, M&A activity.

4 — Prior Context (temporal)
Deal deferred last quarter because champion left. Previous renewal had contentious SLA negotiation. They evaluated a competitor 18 months ago.

5 — Decision Traces
Last contract: VP override for 20% discount (churn risk) → expanded 3 months later. Pattern: exception + positive outcome = expansion signal.

LAYERS 1-3
Mostly available today. Cross-system integration is the challenge, not the concept.

LAYER 4
Partially available. CRM has some history, but the reasoning behind decisions is rarely captured.

LAYER 5
Almost entirely missing. This is the Foundation Capital thesis — and the hardest layer to build.

PRACTICAL EXAMPLE — PART 2

Why Context Changes Everything

The same question — "which opportunities should I focus on?" — gets radically different answers.

Before: ML Scoring Model (Layers 1-2)

```
1. Acme Corp          Score: 87
2. Globex Inc         Score: 74
3. Initech Ltd        Score: 71
4. Wayne Ent          Score: 68
```

A number. No reasoning. No action. The rep looks at it, shrugs, and goes with their gut.

After: Agentic Reasoning (All 5 Layers)

Acme Corp — Expansion opportunity

Usage up 40%. VP of Eng attended last 2 product webinars. They just announced a cloud migration that fits your Platform tier.

Context: Last renewal included a discount exception. Similar accounts expand within 12 months 70% of the time.

Recommended: Personalized email re: their migration, offer Platform trial, loop in SE from original deal.

Nuanced reasoning. Specific actions. Integrated into workflow — email draft, event invite, trial activation.

Agents can handle disparate data types at scale better than humans — but **only if the context is organized, retrievable, and trustworthy.** That's the infrastructure problem.

OPEN PROBLEM 1

How Do You Organize Knowledge and Context?

Today every team writes its own markdowns. Duplicative context, conflicting definitions, no source of truth.

What should the graph architecture be?

The focus is retrieval. The graph is only useful if you can get the right context to the right agent at the right time.

Graph DB (good for traversal) vs. vector store (good for semantic search) vs. hybrid? Schema-first vs. schema-on-read? Entity-centric vs. document-centric indexing?

Should storage be format-agnostic?

Today's dominant pattern: markdown files (CLAUDE.md, cursor rules). But different consumers need different formats.

Proposal: Store in canonical representation, generate delivery format at retrieval time. .md for coding agents, structured docs for RAG chatbots, compact summaries for limited context windows.

Can it be decentralized?

An enterprise-wide graph is a governance nightmare (see every failed Wave 1 & 2 project).

Possibly the only viable path: Start with domain-specific graphs (RevOps, support, engineering). Use entity identifiers as shared keys. Defer cross-domain semantic resolution until a concrete use case needs it.

The architecture question isn't academic — it determines whether context graphs become another failed enterprise data project or actually reach production. **Start narrow, prove retrieval works, federate later.**

OPEN PROBLEM 2

Context Synthesis for Reasoning

Having context in a graph is necessary but not sufficient. How do you get it into an agent's reasoning?

Current pattern: Context7 + MCP

Context7 is an MCP server that injects up-to-date documentation into agent prompts at runtime. When an agent needs context, it queries Context7 via the Model Context Protocol, which pulls versioned docs from official sources directly into the LLM's prompt. Works well for developer documentation — but enterprise context graphs are a harder retrieval problem.

The "context rot" problem

Research shows LLM performance degrades in a U-shaped curve as context grows. More context ≠ better decisions. The field calls this "context rot." The implication: effective agents need the *smallest possible set of high-signal tokens*, not everything in the graph.

Emerging retrieval patterns

GraphRAG

Use graph structure to guide retrieval — follow entity relationships, not just similarity search. Promising but unproven for decision traces at scale.

Agentic retrieval

Let the agent decide what context it needs through iterative queries. More token-efficient, harder to control and evaluate.

Context compilation

Pre-compute contextual summaries for common query patterns. Dense, pre-synthesized context instead of raw traces.

OPEN PROBLEM 3 — THE HARDEST ONE

Governance: The Human Problem

The hardest problems with context graphs are human, not technical.

Problem 1: The undocumented decision ceiling

Humans don't document their reasoning. You can't force them. This has been tried and it fails.

Examples of context never captured:
A deal closed by a phone call with a verbal roadmap commitment. A data warehouse schema chosen for political, not technical reasons. A feature deprioritized because its exec sponsor left. An escalation resolved by a personal favor.

Possible approaches: Instrumented surfaces (capture as byproduct), ambient capture (meetings, Slack, email → LLM extraction), incentive alignment (documentation that helps you), or accepting partial context as a design constraint.

Problem 2: Context deterioration over time

Context rots. Nobody maintains it. The graph silently falls out of sync with reality.

EASIER TO SOLVE
Agent-led workflows with auto-update. CI/CD-generated docs. Event-triggered graph updates (e.g., Salesforce field change).

HARDER TO SOLVE
Annual policy updates with no event. Knowledge in people's heads (and they leave). Definitions that shift by informal consensus. Stale competitive intel.

The real question:
Is there a startup opportunity in making governance *easier* (auto-detect staleness, surface conflicts, assign ownership)? Or does this fundamentally require organizational change that software can't drive?

DISCUSSION

What Should We Build?

Six open problems. One meta-question.

1. Graph Architecture

Entity-centric vs. document-centric. Graph + vector hybrid. Schema design that balances structure with flexibility for retrieval.

2. Decentralized-First

Domain-specific graphs (RevOps, support, eng) that federate later. Entity IDs as the shared key. Defer cross-domain semantics.

3. Context Synthesis

Getting the right context (not all context) to agents. Filtering, formatting, freshness. Solving "context rot" before it kills reasoning quality.

4. Governance Tooling

Auto-detecting stale context, surfacing semantic conflicts, ownership assignment. Making the human problem tractable with software.

5. The Documentation Gap

Reducing friction for humans to capture reasoning. Ambient capture from meetings/comms. Designing for partial context, not completeness.

6. Narrow-First Viability

Where are conditions right today? Regulated domains, system-enforced workflows, areas with structural forcing functions for documentation.

The meta-question: Is context graphs a platform play (picks & shovels for constructing, querying, governing context) or an application play (domain-specific agents that build their own context graph as a moat)? Or both?