DATA ENGINEERING

RAG for Data Engineers

What It Is, Where It Fits, and Why Your Metadata Is the Missing Piece

Conference Session · Data Engineering Track

AGENDA

What We'll Cover Today

01

What RAG actually does under the hood

No hype, just the plumbing

02

Why data teams are uniquely positioned

You already have the context LLMs need

03

4 natural fit points in your workflow

Schema Q&A · SQL agents · Incident response · Tribal knowledge

04

The metadata gap — and how to close it

Practical first steps you can take this month

FOUNDATIONS

The Fundamental Problem with LLMs

What the LLM Knows

- ✓ General SQL syntax
- ✓ Python patterns & idioms
- ✓ Broad data concepts
- ✓ Public documentation

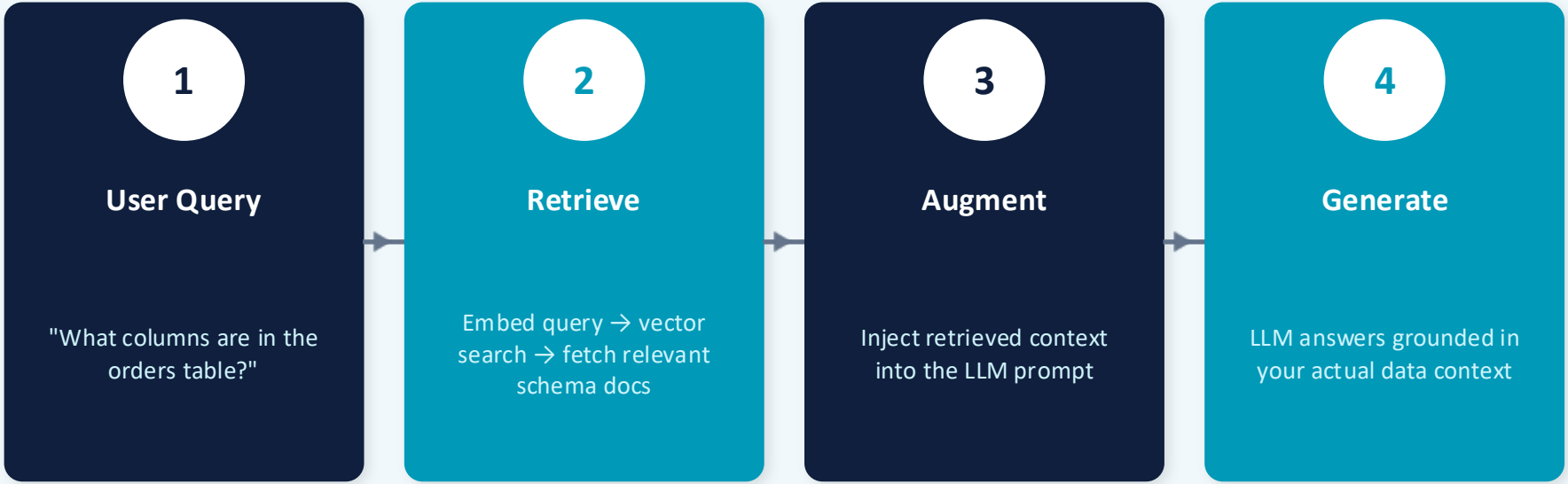
What It Doesn't Know

- ✗ Your table schemas
- ✗ Your lineage graph
- ✗ Your dbt models
- ✗ Your incident history

RAG is the bridge between general intelligence and your specific context.

FOUNDATIONS

How RAG Works (Under the Hood)



Key Insight: The model's weights never change. You're changing what goes into the prompt — dynamically, at query time. This is what makes RAG cheap to operate and easy to update.

YOUR ADVANTAGE

Why Data Teams Are Uniquely Positioned

Structured Schemas

Your table definitions are already machine-readable. That's the exact input format RAG systems love.

Lineage Graphs

You know which tables depend on which. This is gold for answering 'what will break if I change X?'

Query History

Past queries are a rich corpus of intent. What questions have analysts actually asked? You have the answer.

Institutional Docs

dbt model descriptions, Confluence runbooks, Slack threads — structured knowledge waiting to be indexed.

USE CASES

4 Places RAG Fits Naturally in Your Worflow

01

Schema Q&A

Answer catalog questions without a stale data catalog

02

SQL Agents

Ground code generation in your actual table definitions

03

Incident Response

Give on-call agents access to historical pipeline context

04

Tribal Knowledge

Capture what only lives in senior engineers' heads

USE CASE 01

Schema Q&A Without a Stale Catalog

The Pain Today

→ Analyst asks: "what's the grain of orders_fact?"

→ Engineer digs through Confluence page last updated 2021

→ Asks in Slack, waits 2 hours for a reply

→ Finally opens the DDL manually

→ Answer: 15 minutes lost

With RAG

✓ Index your DDL, dbt YAML, catalog descriptions

✓ Analyst asks question in natural language

✓ RAG retrieves the right schema chunk

✓ LLM answers: grain, columns, business rules

✓ Answer: < 5 seconds

USE CASE 02

SQL Agents Grounded in Your Actual Tables

✕ Without RAG

```
SELECT customer_id, total_revenue
FROM customers_table
WHERE created_date > '2024-01-01'
```

Table 'customers_table' does not exist.
Did you mean 'dim_customers'? Column 'total_revenue' not found — it's 'ltv_usd'.

The model hallucinated table and column names because it had no schema context.

✓ With RAG (schema injected)

```
SELECT c.customer_id, c.ltv_usd
FROM dim_customers c
WHERE c.first_order_date
      > '2024-01-01'
```

Query executed successfully. 42,301 rows returned.

RAG retrieved dim_customers DDL and injected it before generation.

USE CASE 03

Incident Response with Pipeline History

02:14 AM	🔴 **Pipeline alert fires — daily revenue model failed**
02:16 AM	→ On-call queries RAG: "revenue model failure Jan 2024"
02:17 AM	→ RAG returns: postmortem from March, similar upstream delay from orders_raw
02:19 AM	→ Engineer checks orders_raw — same late-arriving data pattern
02:23 AM	→ Applies known fix: trigger backfill after 30-min wait window
02:25 AM	→ Pipeline resolves. MTTR: 11 minutes vs. 90-minute previous average

What to index: postmortems · runbooks · Slack incident threads · Airflow task logs · dbt test failures

USE CASE 04

Capturing Tribal Knowledge Before It Walks Out the Door

"Why does the orders pipeline run at 6am and not midnight?"
"Ask Sarah — she designed that 3 years ago to avoid the billing system batch window."

Knowledge Types

· Why tables have certain grain

· Why pipelines run on specific schedules

· Which upstream systems are unreliable

· What data was backfilled and when

· Why columns were deprecated

Sources to Index

· Architecture decision records (ADRs)

· PR descriptions & comments

· Design doc comments & discussions

· Onboarding docs & guides

· Tech spec appendices

How to Capture

· Structured Q&A sessions with senior engineers

· Mine Slack DMs with permission

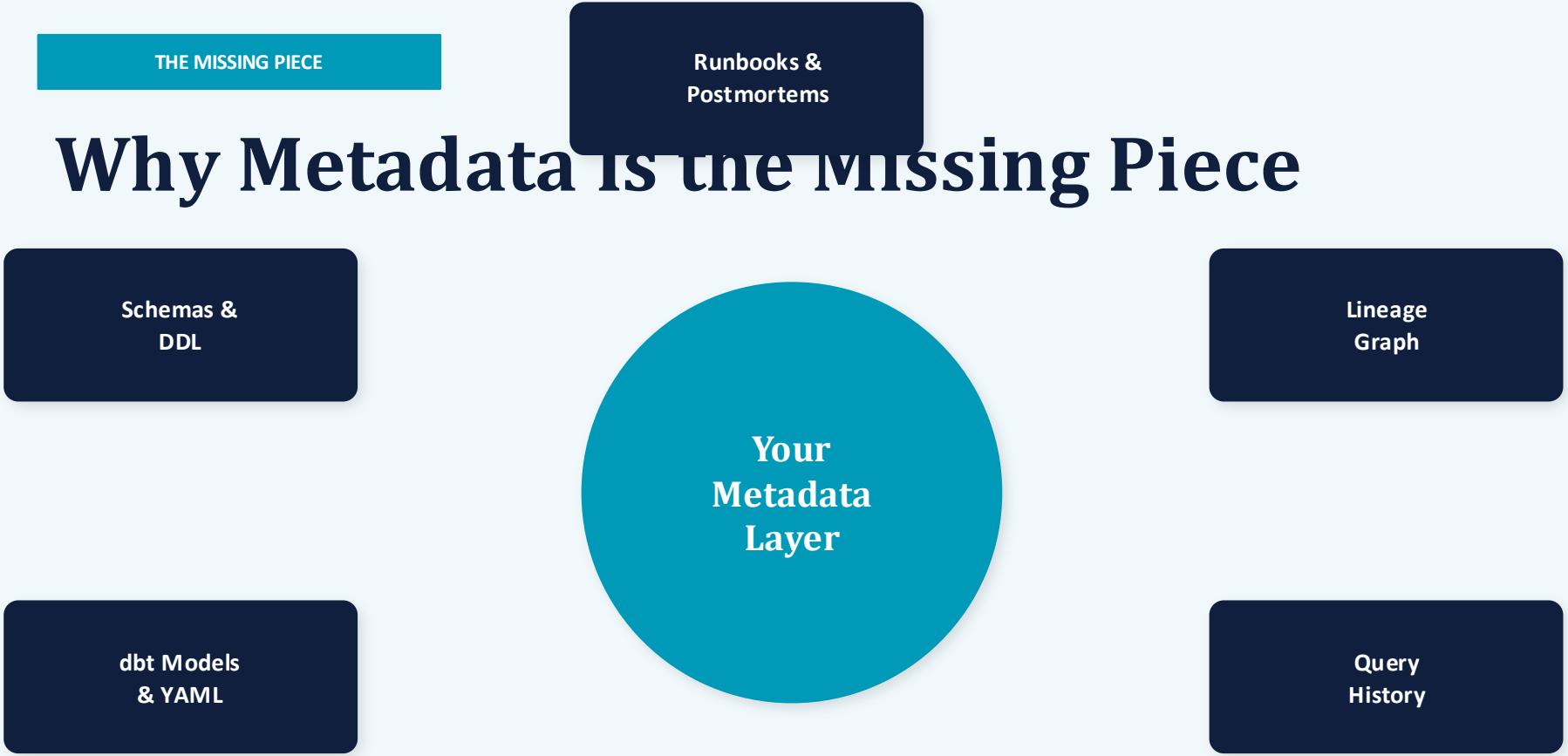
· Extract from architecture review notes

· Annotate existing runbooks

· Generate from code + commit history

THE MISSING PIECE

Why Metadata Is the Missing Piece



Most teams have all of this data. The gap is that it's siloed, not indexed, and not connected to your LLM toolchain. The work is integration, not creation.

ACTION STEPS

Where to Start This Month

Week 1

Audit your metadata assets

List what you have: dbt YAML, data catalog exports, postmortems, ADRs. You need to know what exists before you can index it.

Week 2

Build a minimal schema Q&A

Export dbt manifest.json. Chunk by model. Embed with any open model. Stand up a vector DB (pgvector is fine). Wire up a simple chat interface.

Week 3

Add your first agent use case

Pick SQL agent or incident response based on where your team's biggest pain is. Inject the retrieved context into your agent's system prompt.

Week 4

Measure and socialize

Track: schema questions answered without interrupting engineers. Time-to-resolve on incidents. Show the delta to leadership.

Key Takeaways

1. RAG = dynamic context injection. No retraining. Just better prompts at scale.
2. Data teams already have the raw material LLMs need — schemas, lineage, history, docs.
3. Four natural fit points: schema Q&A, SQL agents, incident response, tribal knowledge.
4. The gap is integration, not creation. This is a data pipeline problem, not an AI research problem.
5. Start with a metadata audit. Build a minimal schema Q&A. Measure the delta. Then expand.

Questions & Discussion

RAG for Data Engineers

What It Is, Where It Fits, and Why Your Metadata Is the Missing Piece

→ What metadata assets does your team currently produce?

→ Where does your biggest on-call pain come from today?

→ What would you build if you had a working schema Q&A in 48 hours?