NetApp

Postgres as an AI Control Plane

Building RAG + MCP Workflows Inside the Database

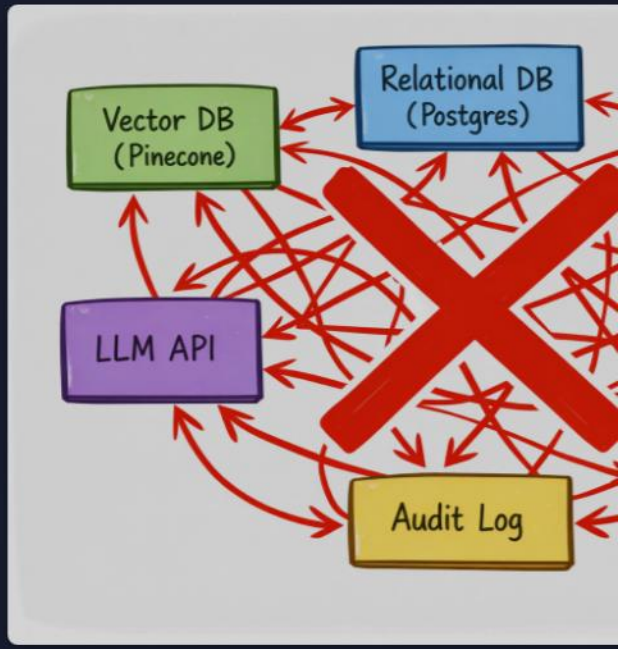
Payal Singh
Senior Database Reliability Engineer
NetApp

Scale23x • March 2026

NetApp

The AI Infrastructure Problem

- Correctness lives in surrounding systems, not the model itself.
- Optimistic AI pipelines assume model correctness; failures surface late and expensively.
- Missing constraints, transactions, and observability leave AI behavior uncontrolled and risk unclear.



NetApp

Why Postgres?

The elephant in the room

NetApp

Why Postgres for AI Workloads

- Postgres acts as the AI control plane, handling deterministic logic and constraints.
- pgvector keeps embeddings inside Postgres with ACID guarantees and standard fea
- One database unifies text, metadata, retrieval, safety controls, and audit logging.

NetApp

Postgres vs Dedicated Vector Databases

Postgres as AI Control Plane

- Vector, full-text, and relational queries in one database transaction
- ACID transactions, backups, replication, audit logging, and row-level security included
- Single stack for text data, embeddings, metadata, and monitoring state

Dedicated Vector DB Tradeoffs

- Often require separate systems for auth, me search
- No built-in SQL joins, transactions, or relatio
- Better fit only when you need billions of vect service

NetApp

Postgres AI Primitives

- pgvector: vector similarity search and high-dimensional embedding storage inside Postg
- pg_trgm: trigram-based fuzzy matching for typo-tolerant, lexical-aware search.
- Built-in full-text search: tsvector/tsquery for lexical retrieval alongside vectors.

PostgreSQL.

The Architecture

How the pieces fit together

The Database IS the Control Plane

Postgres is the deterministic control plane; t
LLM is a constrained, stochastic worker.

The database does everything deterministic; the LLM only drafts and revises under constraint.

NetApp

Schema Design & Security

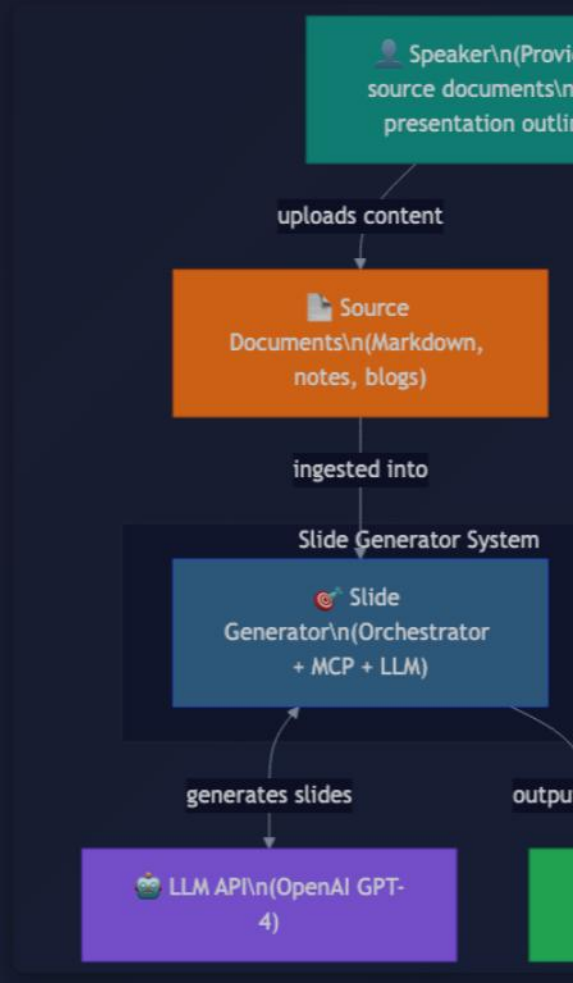
- SECURITY INVOKER with pinned search_path enforces caller privileges and prevents functio
- REVOKE PUBLIC with least-privilege GRANT lets only the app role run specific functions.
- Append-only audit tables and typed MCP functions ensure immutable logs and validated, no

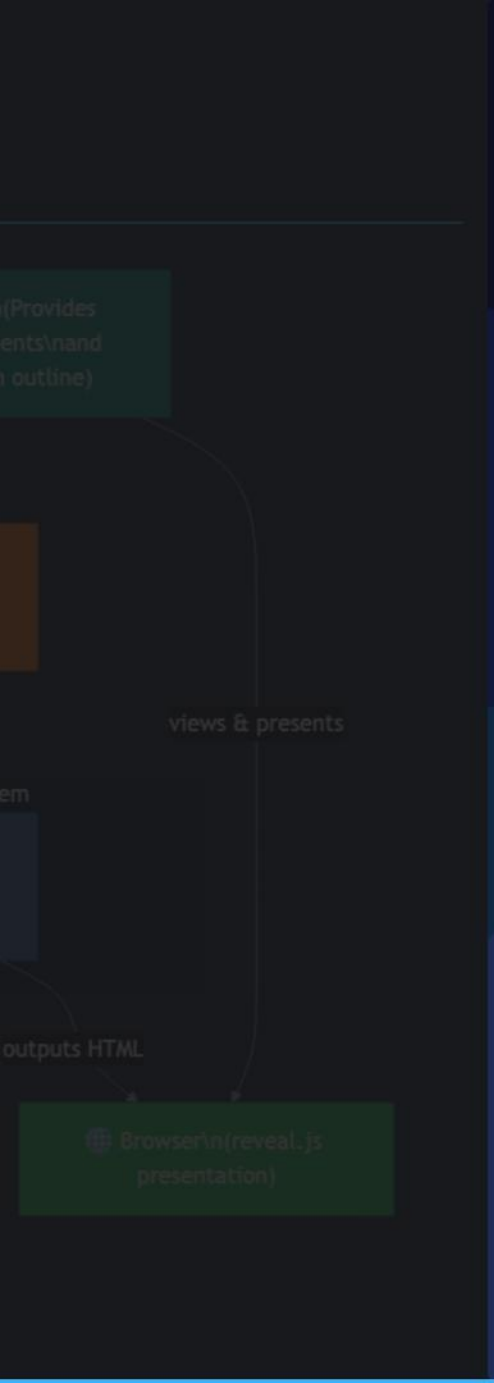
NetApp

System Architecture

- Postgres as AI control plane
- MCP tools, not raw SQL
- LLM orchestrator drives workflow

End-to-end slide generator: Postgres-centered control plane with MCP, orchestrator, and renderer.





NetApp

RAG & MCP

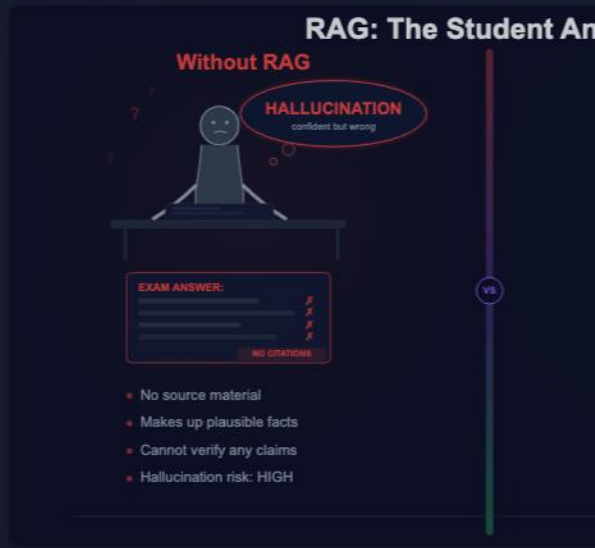
Under the hood

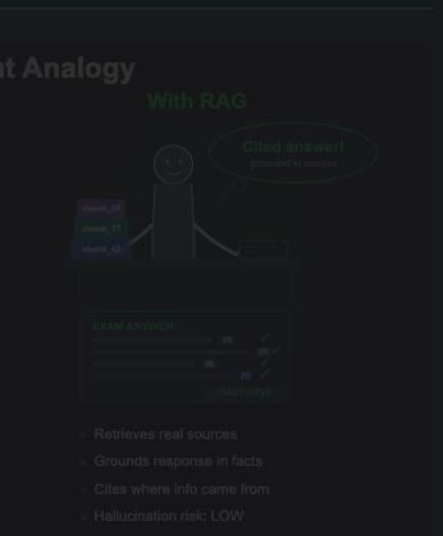
NetApp

What is RAG?

- Retrieve before you generate
- Ground answers in your data
- Hybrid search, not just vectors

RAG retrieves relevant context, augments the prompt, then the LLM generates grounded answers.





RAG Inside Postgres

```
WITH sem AS (
  SELECT c.chunk_id id, 1-(c.embedding <=> :qv) s,
         ROW_NUMBER() OVER (ORDER BY c.embedding <=> :qv) r
  FROM chunk c WHERE c.embedding IS NOT NULL LIMIT 20
), lex AS (
  SELECT c.chunk_id id, ts_rank_cd(c.tsv, plainto_tsquery(:q)) s,
         ROW_NUMBER() OVER (ORDER BY s DESC) r
  FROM chunk c WHERE c.tsv @@ plainto_tsquery(:q) LIMIT 20
)
SELECT coalesce(sem.id, lex.id) chunk_id,
       :sw/(:k+coalesce(sem.r,41))+:lw/(:k+coalesce(lex.r,41)) rrf
FROM sem FULL OUTER JOIN lex USING (id)
ORDER BY rrf DESC LIMIT 10;
```

Hybrid query combines pgvector semantic and tsvector lexical search via RRF.

RRF merges rank positions from both arms, avoiding score normalization issues.

NetApp

Beyond Vector Search: Two-Stage Retrieval

Stage 1: Hybrid RRF in Postgres

- Hybrid retrieval: pgvector semantic + tsvector lexical in one SQL query, using cosine distance with HNSW and ts_rank_cd with GIN.
- RRF combines ranks: 1/(k + semantic_rank) + 1/(k + lexical_rank), avoiding mixed-score scaling issues.
- Postgres stage is recall-optimized, casting a wide net in under ~20ms.

Stage 2: Cross-Encoder Rerankin

- Python cross-encoder reranks top-K candid
  using cross-encoder/ms-marco-MiniLM-L6
  chunk scoring.
- Cross-encoder jointly attends to query and
  delivering higher precision than bi-encoder
- Latency ~50–100ms for ~50 query–chunk
  with Stage 1 keeps total retrieval under ~100

NetApp

nking

andidates from Postgres
M-L6-v2 for joint query–

and passage tokens,
oder embeddings alone.

hunk pairs on CPU; combined
~100ms.

What is MCP?

- Open protocol "USB-C" for AI
- Typed tools, not raw SQL
- Clear safety boundary

MCP connects LLM apps to Postgres tools through a safe, typed client-server boundary.

MCP Boundary

SAFE: Typed MCP Tools

Predefined function signatures

Input validation (Pydantic)

SECURITY INVOKER functions

Least-privilege access

Full audit logging

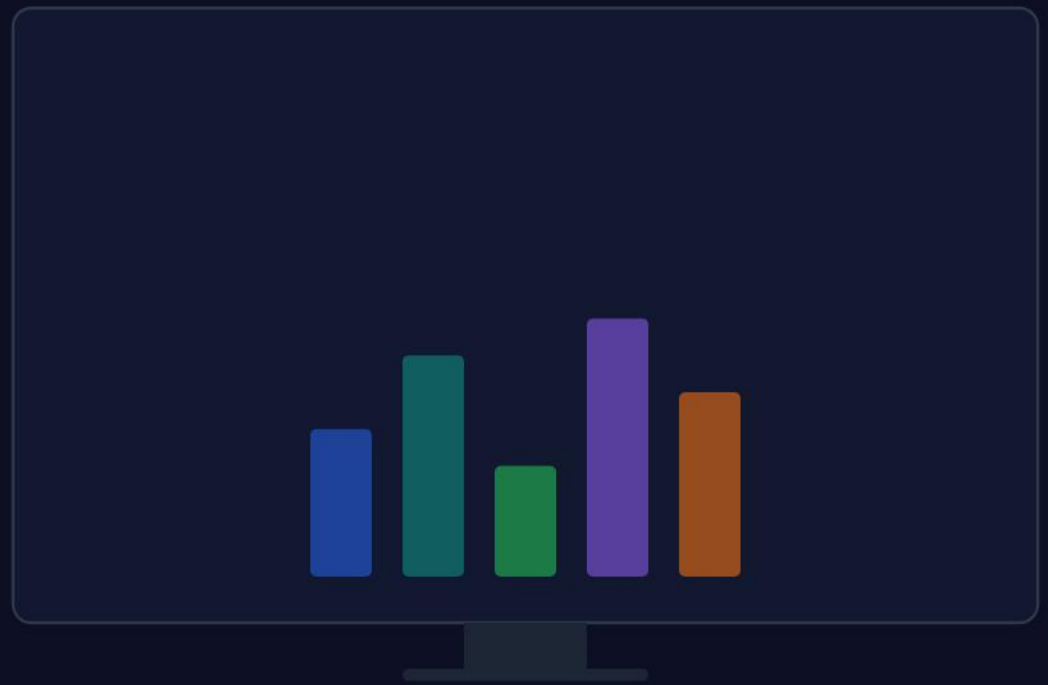
NetApp

Typed Tools, Not Raw SQL

```
// # knowledge tools
async function mcp_search_chunks(query: string, topK: number) {}

// # deck orchestration
async function mcp_pick_next_intent(deckId: string) {}

// # validation gates
async function mcp_validate_slide_structure(slideSpec: Json) {}
async function mcp_check_grounding(deckId: string,
                                   slideSpec: Json) {}

// # commit
async function mcp_commit_slide(deckId: string,
                                slideNo: number,
                                slideSpec: Json) {}
```

MCP tools are typed wrappers over Postgres functions, never raw SQL.

Each tool maps to a specific DB function category: search, validate, commit.

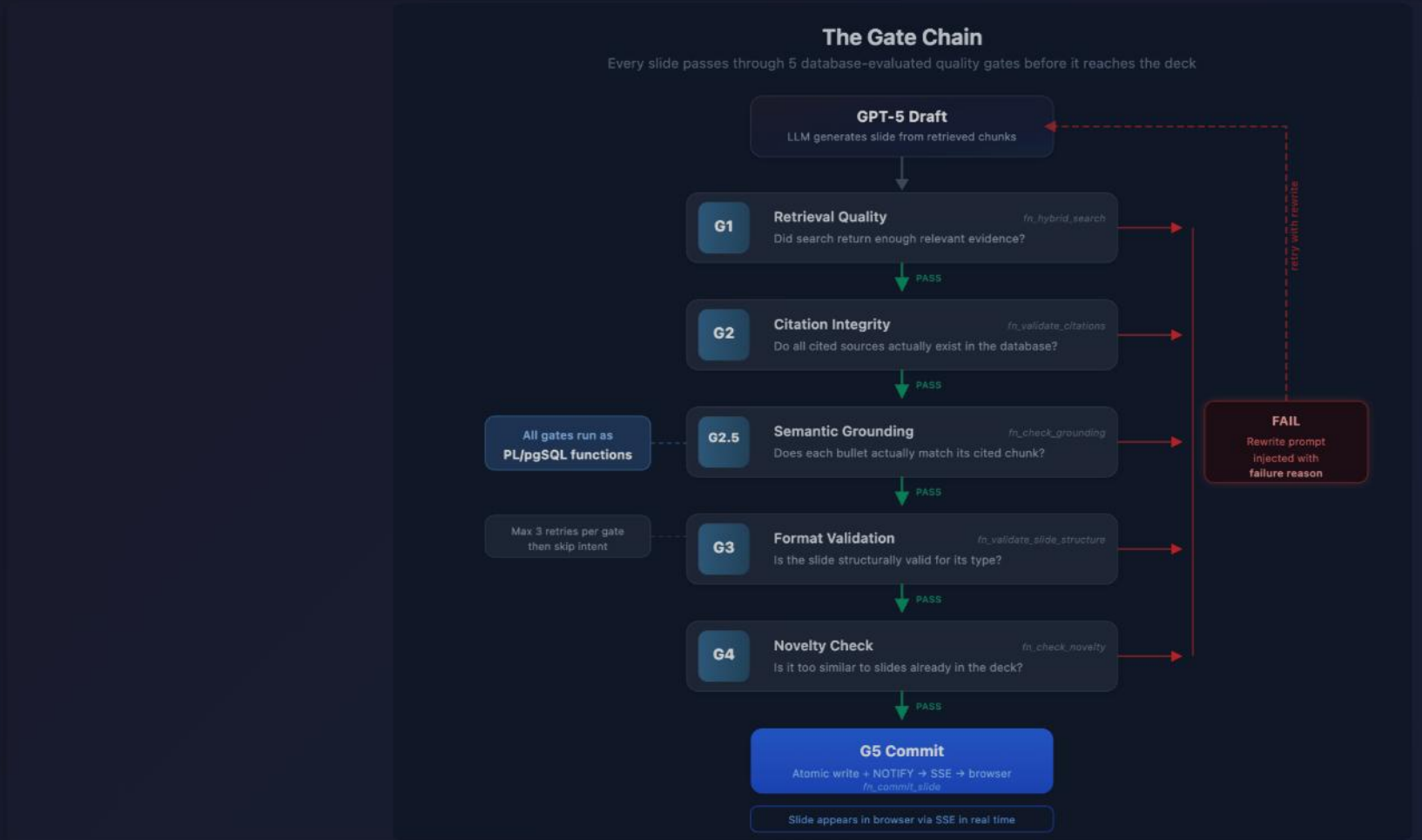


Control & Observability

Making the invisible visible

NetApp

Control Gates & Validation



NetApp

Observable AI with SQL

```
SELECT gate_name,
       decision,
       reason,
       COUNT(*) AS occurrences,
       ROUND(AVG(score)::numeric, 3) AS avg_score
FROM gate_log
JOIN generation_run USING (run_id)
WHERE deck_id = $1
GROUP BY gate_name, decision, reason
ORDER BY occurrences DESC;
```

Aggregate gate outcomes per deck to see which checks fail most often.

Immutable gate_log entries make these observability queries non-repudiable.

NetApp

The Demo

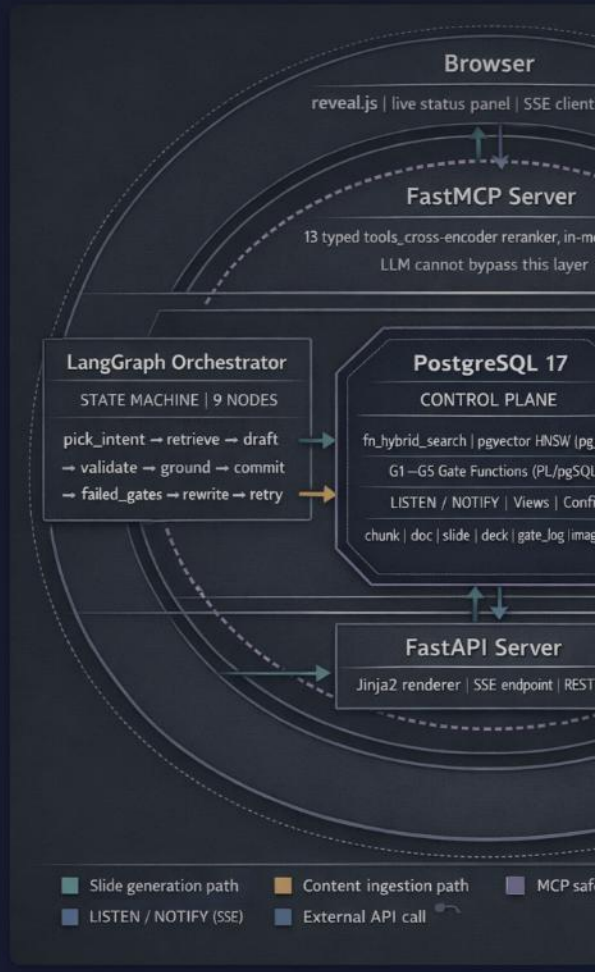
Live from PostgreSQL

NetApp

What We Built

- Postgres-first slide generator
- End-to-end auditable pipeline
- These slides are the demo

Diagram: docs → chunks → gates → slides, all governed and logged inside Postgres.



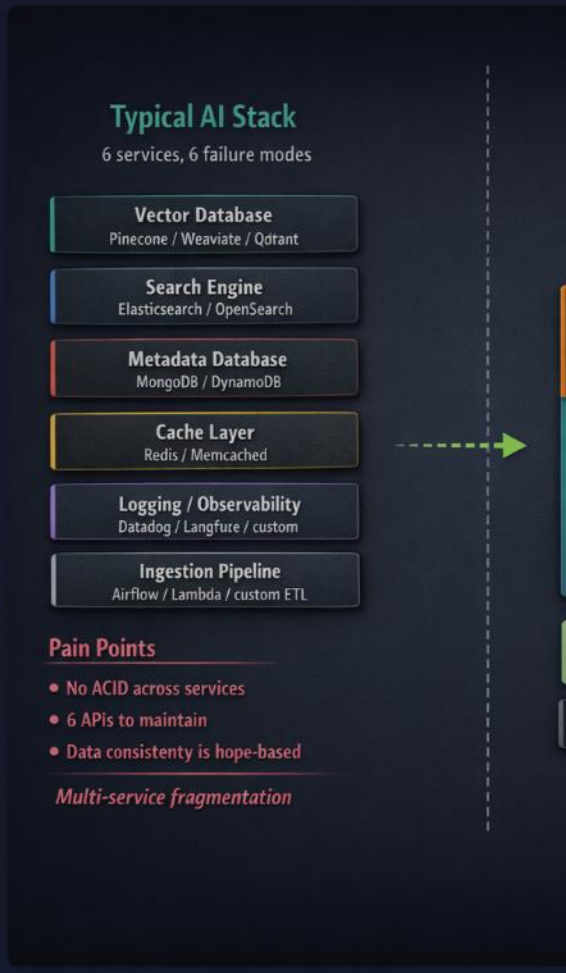
Key Takeaways

Core Ideas

- Postgres is the AI control plane, not just long-term storage.
- RAG here means retrieval + state + gates + provenance.
- MCP tools form the safety boundary instead of exposing SQL.

What Postgres Handles

- Hybrid retrieval, safety gates, and state management all run inside Postgres.
- Postgres verifies grounding, logs decisions, and enforces security boundaries.
- The demo shows Postgres unifying retrieval, safety, and orchestration.



NetApp

Thank You & Questions

- GitHub: github.com/payals/pg_rag_slide_generator
- LinkedIn: linkedin.com/in/payalsingh
- Email: payal.singh@netapp.com
- Slides: Generated by this very system!