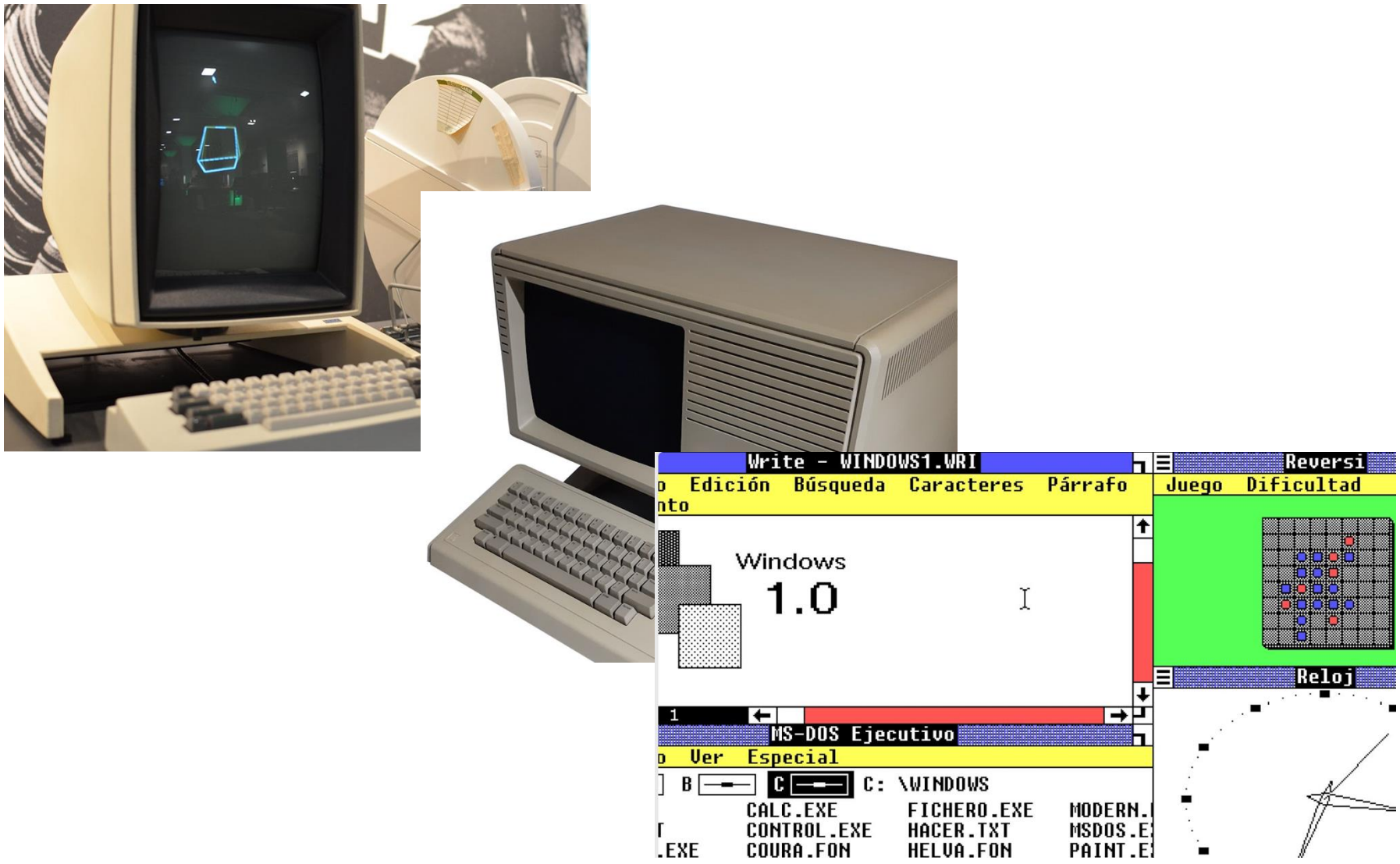
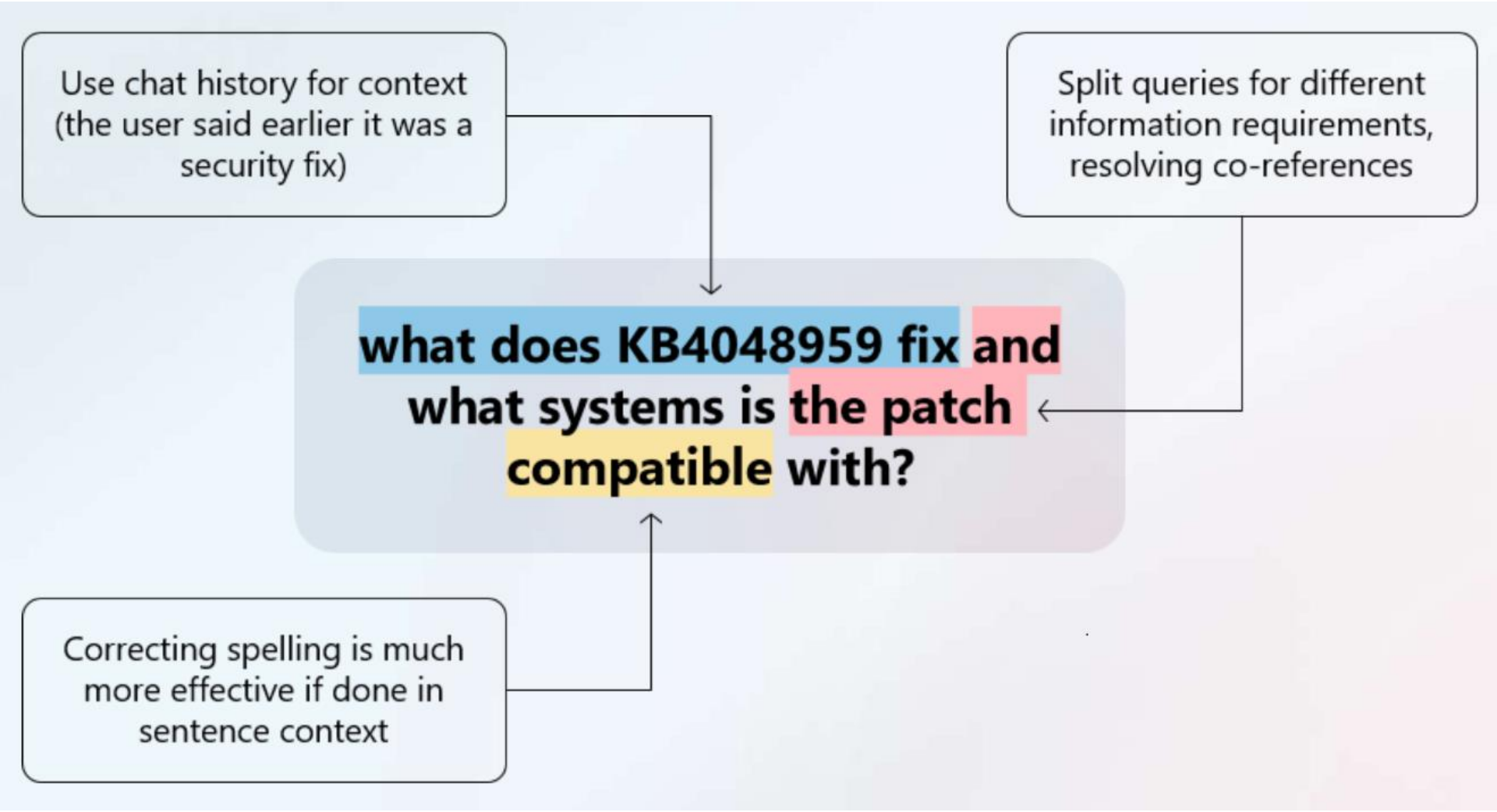
How Agentic RAG Solves Naïve RAG Weaknesses

Based on RAG Weak Points Tables – Ricoh, RNA Meeting July 2025

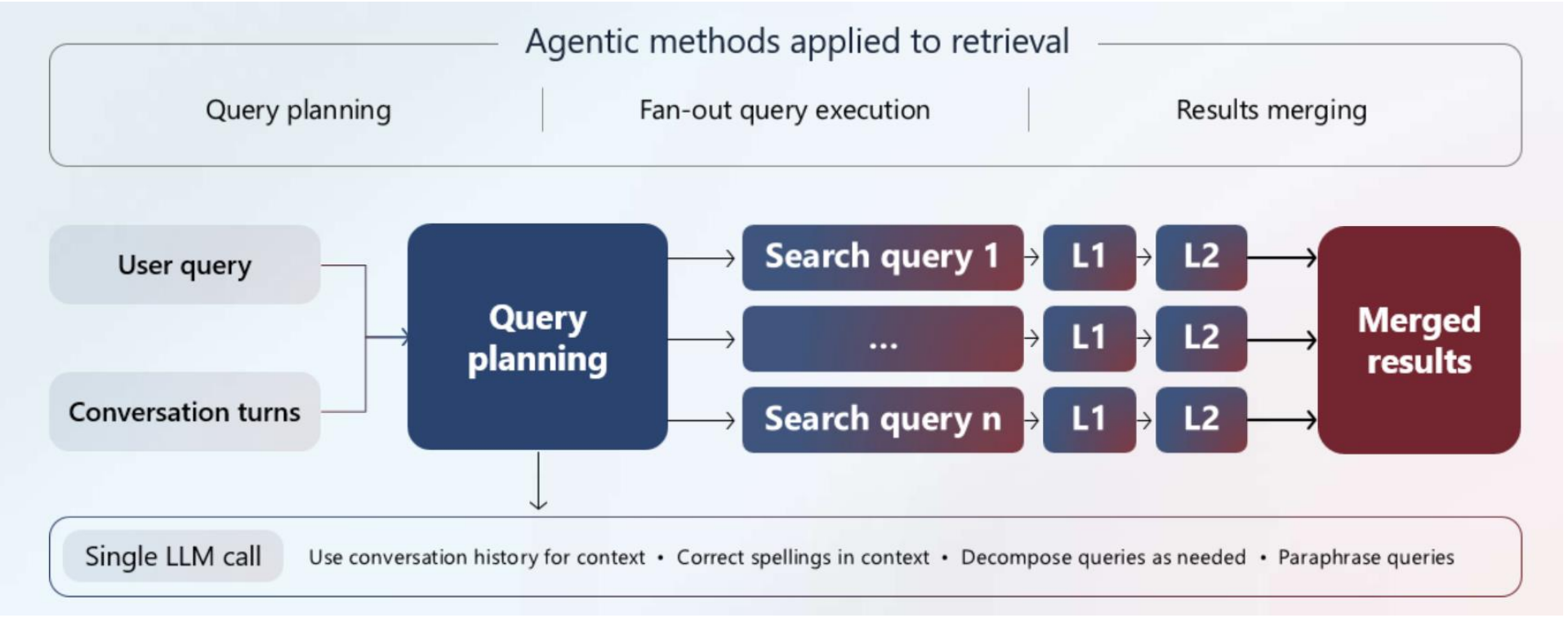
Xerox PARC visit moment



Example


Use chat history for context (the user said earlier it was a security fix)
Split queries for different information requirements, resolving co-references
what does KB4048959 fix and what systems is the patch compatible with?
Correcting spelling is much more effective if done in sentence context

Architecture



General Naïve RAG Failure Modes

Weak spot	What goes wrong	Concrete example
Compound / multi-hop questions	A single chunk rarely contains all required facts; similarity search returns pieces that answer only part A or part B, so the LLM hallucinates or guesses.	"Who led the ServiceNow rollout and which cost-saving KPI did the project hit in its first quarter?"
Comparative / option evaluation	Retrieval brings chunks about each item in isolation; the model may produce an ungrounded opinion or miss a dimension.	"Should we host the vector DB on Cosmos DB vs. Pinecone? List trade-offs for latency, price, DevOps effort."
Long-range summarization & synthesis	Large corpora exceed the context window; sampling yields an incomplete picture.	"Give me the 5-bullet executive summary of all incident-post-mortems from the last fiscal year."
Numeric aggregation & filtering	Chunks store raw numbers; LLMs struggle with arithmetic over many dispersed values.	"Total downtime minutes for StoreNext clusters across all 2024 incidents—broken down by root cause."
Temporal reasoning	Vector similarity ignores time; older docs outrank fresher ones unless filtered.	"What are the latest security patches still pending on production?"
Domain-specific jargon / synonyms	Embeddings may not capture subtle synonymy; the right doc stays below top-k.	Query: "steps to fail over the telephony core" → top hits cover PBX firmware updates, but not the DR run-book.

HR Knowledge Base – Naïve RAG Gaps

Weak spot	What goes wrong	Concrete example
Compound / multi-hop questions	Top-k retrieval brings either the policy or the procedure, not both; LLM guesses the missing half.	"How many vacation days can I carry over and which form do I submit to request it?"
Comparative / option evaluation	Chunks describe each benefit plan in isolation; no doc explicitly contrasts them.	"Compare premium costs and coverage between the BlueShield PPO and the Kaiser HMO plans."
Long-range summarization & synthesis	Engagement-survey results span dozens of PDFs; only a few get stuffed into the prompt.	"Give me a three-bullet summary of the 2024 Employee Engagement Survey across all regions."
Numeric aggregation & filtering	Totals must be computed across multiple policies; LLM flubs arithmetic.	"How many paid holidays do US-based employees get on average versus Japan-based employees?"
Temporal reasoning	Vector search returns older but semantically similar docs unless date-filtered.	What changes were made to parental-leave policy after 2023?
Domain jargon / synonyms	Embeddings miss Ricoh-specific terms, so the right doc falls outside top-k.	I need to take some R&R days, what forms should I complete to request LOA next month?

General Naïve RAG Failure Modes vs Agentic RAG

Naïve Weakness	What Goes Wrong	Agentic RAG Fix
Compound / Multi-hop Questions	Only retrieves part of the answer; LLM guesses the rest.	Agent decomposes question and retrieves for each sub-part.
Comparative / Option Evaluation	Retrieves isolated facts; lacks structured comparison.	Agent compares retrieved content explicitly, may use tools.
Long-range Summarization	Incomplete context from large corpora.	Uses map-reduce (Hadoop) summarization across batches of docs.
Numeric Aggregation & Filtering	LLMs fail at multi-chunk arithmetic.	Agent calls external tools (e.g., SQL, Pandas) to compute.
Temporal Reasoning	Older documents dominate due to vector similarity.	Agent filters and prioritizes by metadata like date.
Domain Jargon / Synonyms	Misses internal terms; wrong chunks retrieved.	Agent reformulates queries and expands synonyms.

Agentic RAG Fixes – HR Examples

Examples from HR knowledge base:

- Vacation days + form → Agent splits into policy + procedure lookup.
- PPO vs HMO → Agent builds side-by-side comparison.
- Survey summary → Map-reduce summarization over all regions.
- Holiday averages → Uses arithmetic tools to aggregate by country.
- Parental leave after 2023 → Filters results using metadata.

Mitigation techniques

1. **Iterative (multi-hop) retrieval** – decompose the user query into sub-questions; retrieve for each hop; chain the answers.
2. **Reranking & fusion-in-decoder** – pull a wider candidate set (*k* = 50-100), rerank with a cross-encoder, then weave snippets together during generation.
3. **Hierarchical or map-reduce summarization** – summarize batches of chunks, then summarize the summaries.
4. **Structured tool calls** – push arithmetic, filtering, or comparison to an external tool (SQL, Pandas) and feed results back to the LLM.
5. **Metadata-aware retrieval** – store and query by tags like “date,” “system,” or “owner,” not just dense vectors.
6. **Agentic orchestration** – use planners (e.g., ReAct, GraphRAG) that can decide “I need more context for part B—run another search.”

Summary: Techniques Used by Agentic RAG

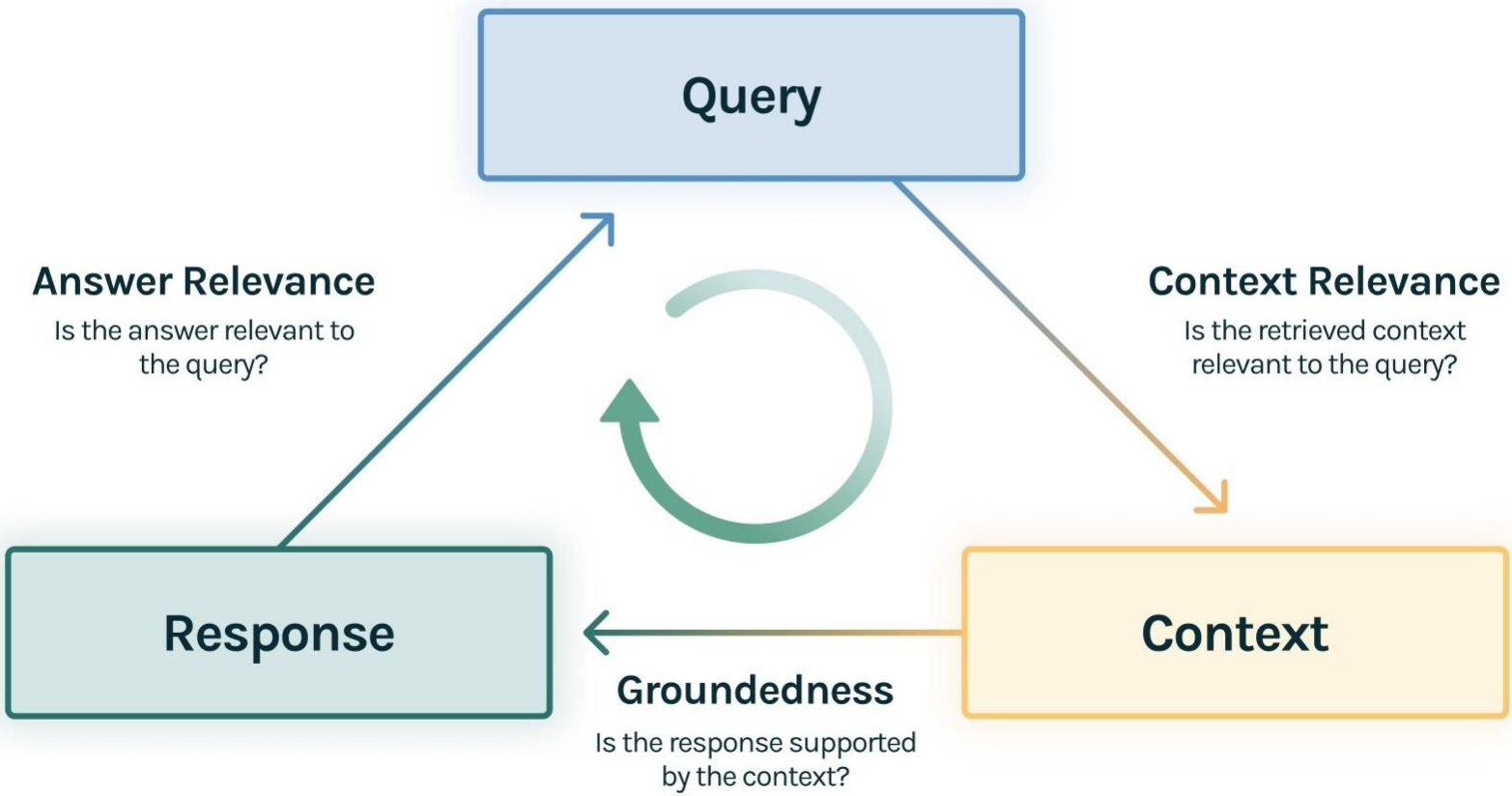
Agentic RAG combines multiple advanced techniques:

- Multi-hop planning with ReAct / GraphRAG
- Metadata filtering (e.g., by date, region)
- Structured tools for math & filtering (SQL, Pandas)
- Hierarchical summarization (map-reduce style)
- Cross-encoder reranking and fusion-in-decoder
- Reflective retry if answer is incomplete

Naïve/Simple RAG vs Agentic RAG

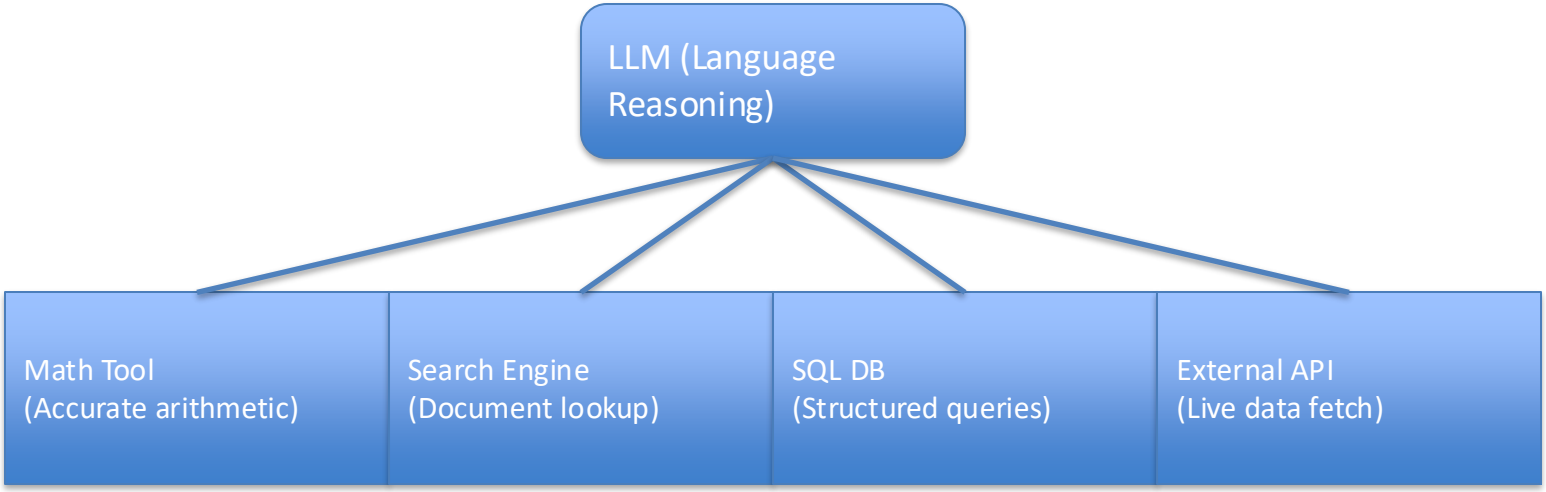
Feature	Naïve RAG	Agentic RAG
Retrieval	Static query to vector DB	Multi-hop, dynamic tool usage
Context handling	Fixed chunks	Context-aware reasoning
Execution	Single-pass	Iterative planning & refinement
Integration	Standalone answer	Workflow automation & action execution

The RAG Triad



https://www.trulens.org/getting_started/core_concepts/rag_triad/

MRKL: Modular Reasoning with Knowledge and Language



The MRKL pattern uses an LLM to route sub-tasks to specialized tools (math, search, SQL, APIs), then synthesizes the outputs into a final, coherent answer.

Key takeaway

- A naïve "retrieve-top-k-then-stuff" pipeline is great for *local fact lookup* but brittle for **reasoning that spans documents, options, or time**.
- Layering smarter retrieval (filters, rerankers), structured reasoning, and multi-step orchestration transforms RAG from a FAQ helper into a robust enterprise knowledge system.

Agentic Retrieval Concepts

Key Insights from Microsoft Learn

Agent = Reasoning + Memory + Action

- Agents do more than retrieve:
- Reason about the user’s intent.
- Maintain memory of previous steps and gaps.
- Take action: search, calculate, call tools, refine queries.

Example: For Q2 metrics, the agent retrieves data, runs calculations, and revises the output as needed.

Orchestration Patterns for Agents

- Agentic RAG uses intelligent workflows:
- ReAct: Combines reasoning and tool use (think-act-think).
- MRKL: Modular Reasoning with tools for math, APIs, and search.
- GraphRAG: Uses a knowledge graph for semantic search and traversal.

Structured Retrieval = Higher Precision

- Agents filter and rank with metadata:
- Use fields like doc_type, project_id, or date_updated.
- Improves results for time-sensitive or domain-specific queries.
- Avoids reliance on dense vectors alone.

Example: Filter to docs updated in the last 30 days to answer 'What’s still pending?'

Tool Use is First-Class

- Agents don’t just guess—they use real tools:
- SQL queries, APIs, Excel-like computation tools.
- Enables accurate math, comparison, and dynamic lookup.
- Supports grounded enterprise logic.

Example: To compare sprint velocity, the agent queries Jira and calculates rankings.

Memory Powers Intelligent Retrieval

- Agents keep track of what's known or missing:
- Maintains short-term memory of retrieved and used documents.
- Prevents repetition, hallucination, and context gaps.
- Enables long-form synthesis across steps.

Agent ≠ Just an LLM

- Agents orchestrate, LLMs assist:
- Agent decides when and how to retrieve.
- LLM generates queries, parses output, or writes summaries.
- Tools and APIs provide facts, calculations, and results.
- Enhances traceability, security, and flexibility.

Final Takeaway

- Agentic RAG = Think + Plan + Act
- Not just Q&A – it’s structured knowledge work.
- Dynamically adapts to user needs.
- Uses tools, memory, filters, and multi-step logic.
- Ideal for robust enterprise-scale answers.

Resources

- https://www.youtube.com/watch?v=Fjef7uhV3gw
- https://learn.microsoft.com/en-us/azure/search/search-get-started-rag?pivots=python
- https://learn.microsoft.com/en-us/azure/search/search-agentic-retrieval-concept
- https://www.youtube.com/watch?v=PeTmOidqHM8
- https://www.trulens.org/getting_started/core_concepts/rag_triad/

- https://github.com/Azure-Samples/azure-search-python-samples