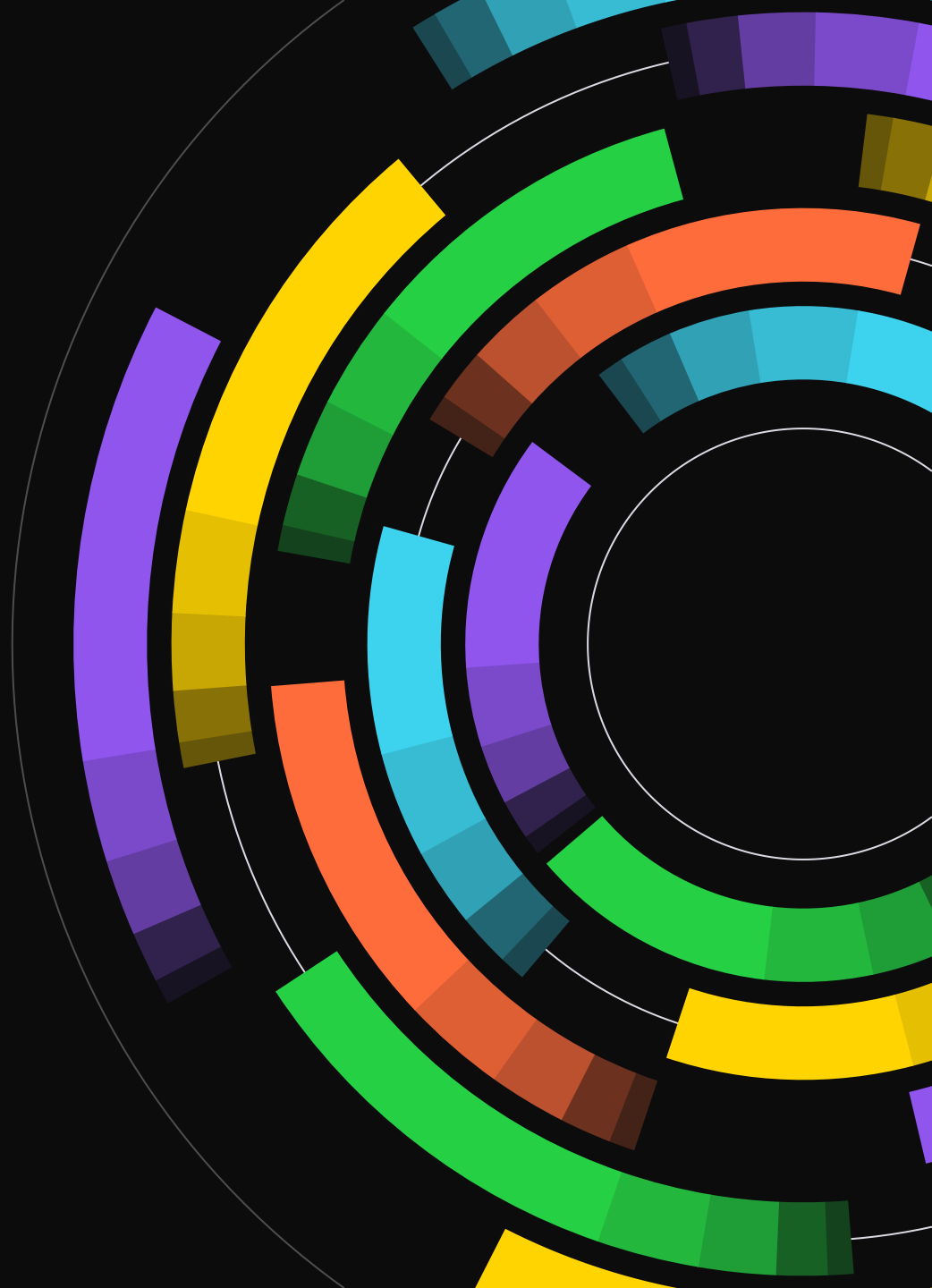
mariaDB

Look to AI for Your Financial Reporting Needs in an AI world

Kathy Gibbs

Manager, NAMER SE

November 5, 2026



Agenda

#	Topic
01	Who am I?
02	Increasing your AI readiness
03	Using vector search
04	Why RAG helps
05	The MariaDB AI Platform

MariaDB Confidential

Who am I?

Kathy Gibbs, Manager, NAMER SE

Over 30 years in relational databases, including Oracle, SQL Server, MySQL, MariaDB and PostgreSQL

Worked in retail, public sector, finance and construction, and for companies like Oracle, AWS, SolarWinds and EDB

Now focused on using AI techniques with relational databases, so DBAs finish routine tasks quickly and get to the “fun” work



02

02 / The AI landscape

AI readiness

The problem is the Frankenstein stack

Legacy architectures can’t keep pace with AI agents and workflows

A fragmented stack

- Different people use different tools
- Old tech forced into a new process
- Separate vector databases and complex ETL pipelines
- Security risks, slow and complex

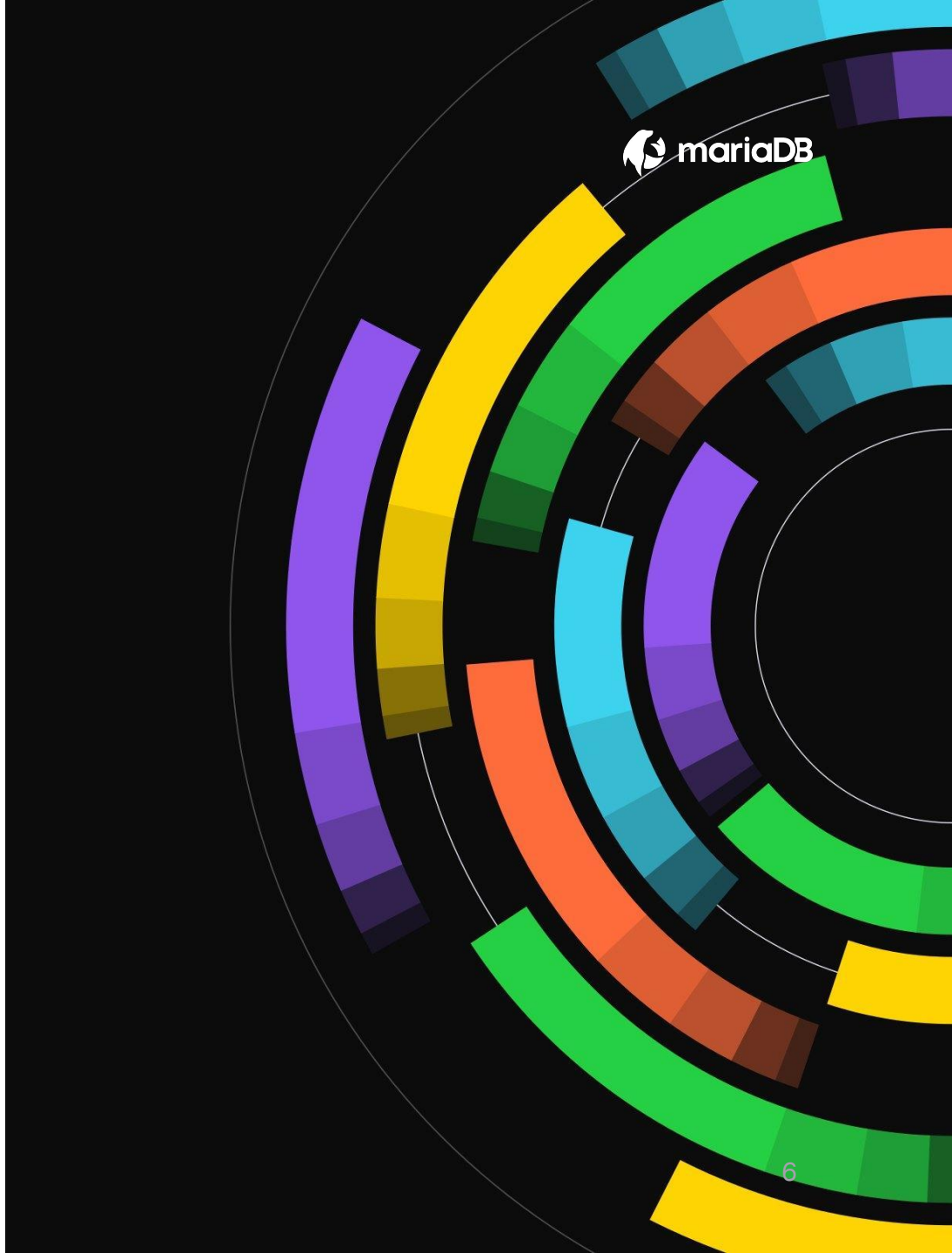
Latency is key

- AI agents need data at speed
- Reporting on data in microseconds
- Think fraud detection and reaction

What is AI readiness?

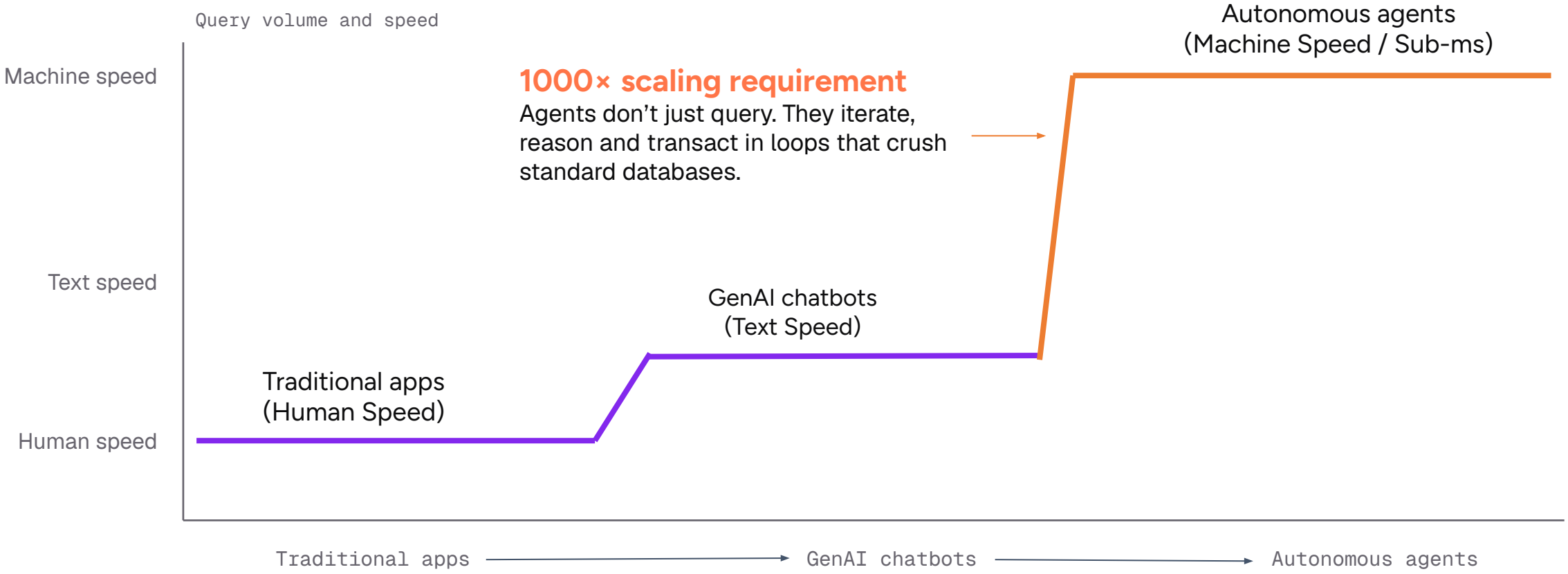
From simple text generation to autonomous digital labor

- The AI inflection point marks the shift from AI as a chatbot to an agentic structure
- Autonomous systems reason, plan multi-step work with agents and execute complex tasks with real-world consequences
- We are moving from simple text generation to autonomous digital labor that actually runs business processes



The AI inflection point

Traditional architectures cannot support the velocity AI workflows need



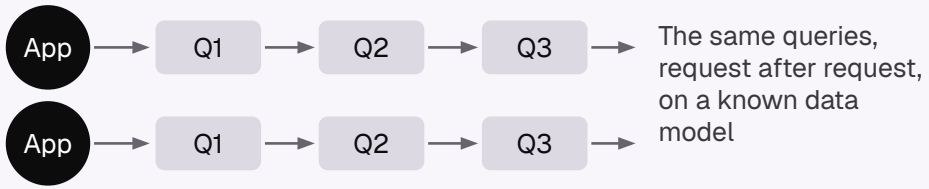
MariaDB Confidential

Agents don't use financial data the way applications do

Infrastructure was built for fixed, tuned queries. Agents write theirs at runtime.

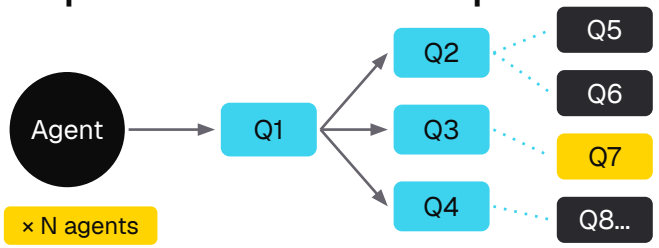
AN APPLICATION

A known set of queries



AN AGENT

Unpredictable runtime queries with no fixed pattern



No fixed query sequence
Queries branch dynamically as results dictate next steps.

Unknown depth and count
Workload volume cannot be tuned in advance.

Multiplies exponentially across agents

Property	An application	An agent	So the database needs to
Who writes it	A developer, ahead of time	A model, at runtime	Understand any query, even ones nobody wrote
Variety	A known set	Open ended	Respond very fast without tuning
What it triggers	A fixed next step	Several queries, each can branch	Remember and reuse answers, so responses stay fast
Workload	One, known in advance	Often several in one task	Hold any data type in one place
Review	Reviewed and tuned	Unreviewed, retried	Be secure by design, in one simple setup
Concurrency	Pooled, same shapes	Many agents and branches	Handle bursty load at 1000x scale
Data it generates	Business records, session state	Business records, its own memory, checkpoints and handovers	Keep agent data apart from business data

Vector search

How vector search works

HNSW is a graph algorithm for approximate nearest-neighbor search

HNSW builds a graph in which each vector is a node and edges connect it to other vectors.

HNSW runs in memory. Saving the index for later use requires a separate storage mechanism.

Distance calculations are the most performance-sensitive part of HNSW, accounting for an estimated 80–90% of search time.

To rank matches, the search only needs to know which vectors are closer to the query vector.

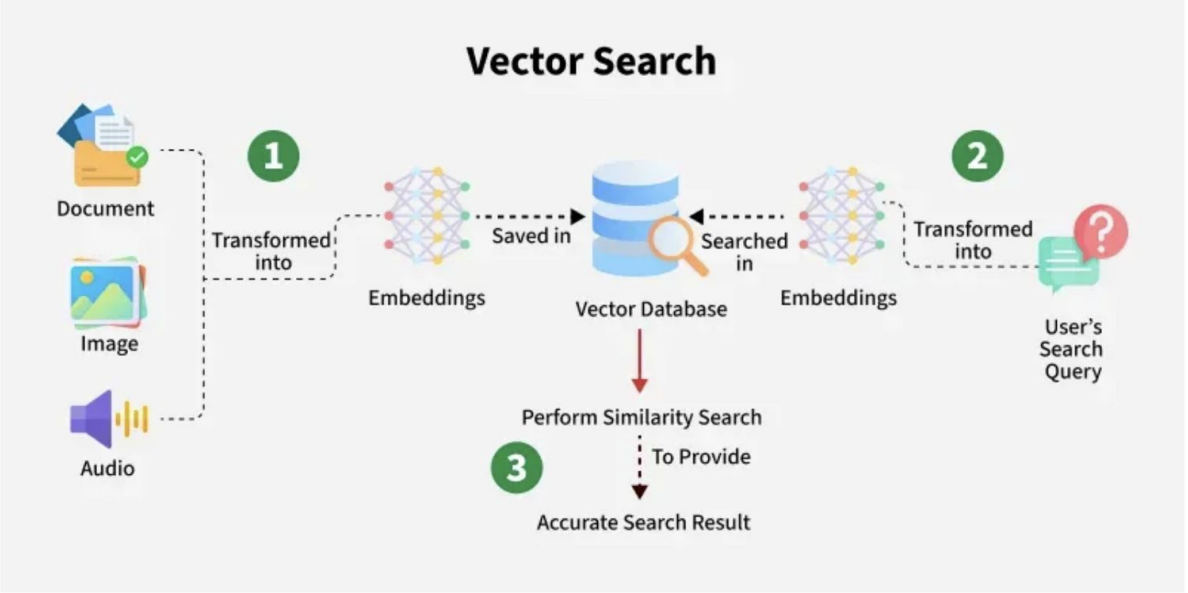


Image from geeksforgeeks.org

Optimize distance calculations

For high-dimensional vectors and Matryoshka embeddings

Matryoshka embeddings

- Many components, ranging from dozens to tens of thousands
- Pack several smaller embeddings inside a single vector, like Russian nesting dolls
- Models store the most critical information in the earliest dimensions
- Shorter slices cut storage and speed up search while retaining much of the original information

Native in MariaDB Server

- No extension or separate vector database required
- On-premises and on Amazon RDS for MariaDB
- Embeddings live in a `VECTOR` column alongside relational data
- Combine vector similarity search and SQL filters in one transactional statement

Vector high-dimensionality

From dozens to tens of thousands of components per vector

Number of features

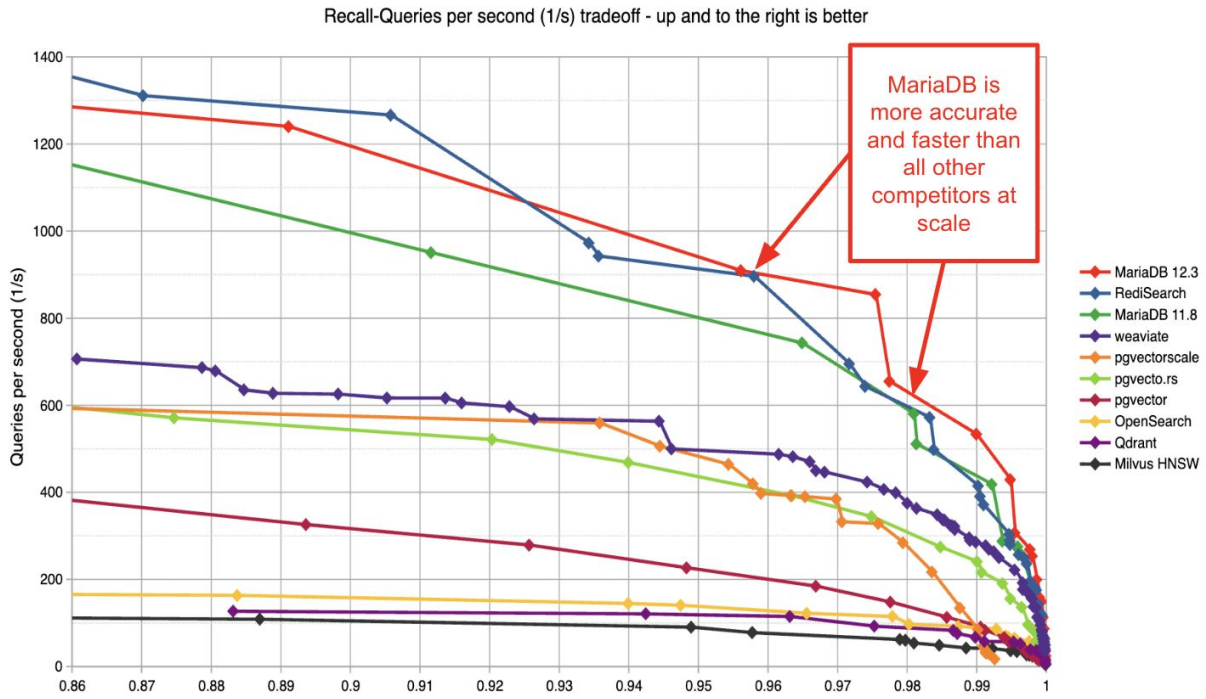
A vector represents the features of an entity. A 1,000-dimensional vector measures 1,000 distinct attributes for one data point, such as the weight of 1,000 vocabulary words in a document or thousands of pixel color values in an image.

Orthogonality and hyperspaces

In thousands of dimensions, vectors such as LLM embeddings have enough room to become perpendicular to one another. That helps algorithms tell very different concepts apart without semantic overlap.

MariaDB vector search

Superior performance, higher accuracy, faster execution at scale



Source: 2026 MariaDB internal benchmarks & product documentation · mariadb.com

Faster execution

Outpaces competitors in speed and throughput

- **pgvector** 3.4x to 4x QPS (~240%–300% faster) at 94% recall
- **Weaviate** ~1.5x to 2.1x QPS (~50%–110% faster) at 94% recall

Vector native

Vectors and SQL in a single statement

- **Runs in-engine** alongside transactional and JSON data
- **No 2nd database or sync pipeline** required

Why it matters

- Eliminates the trade-off between search accuracy and query speed
- High recall provides relevant context, reducing AI errors
- Simplifies architecture without extra vector DBs or caches

04

04 / RAG

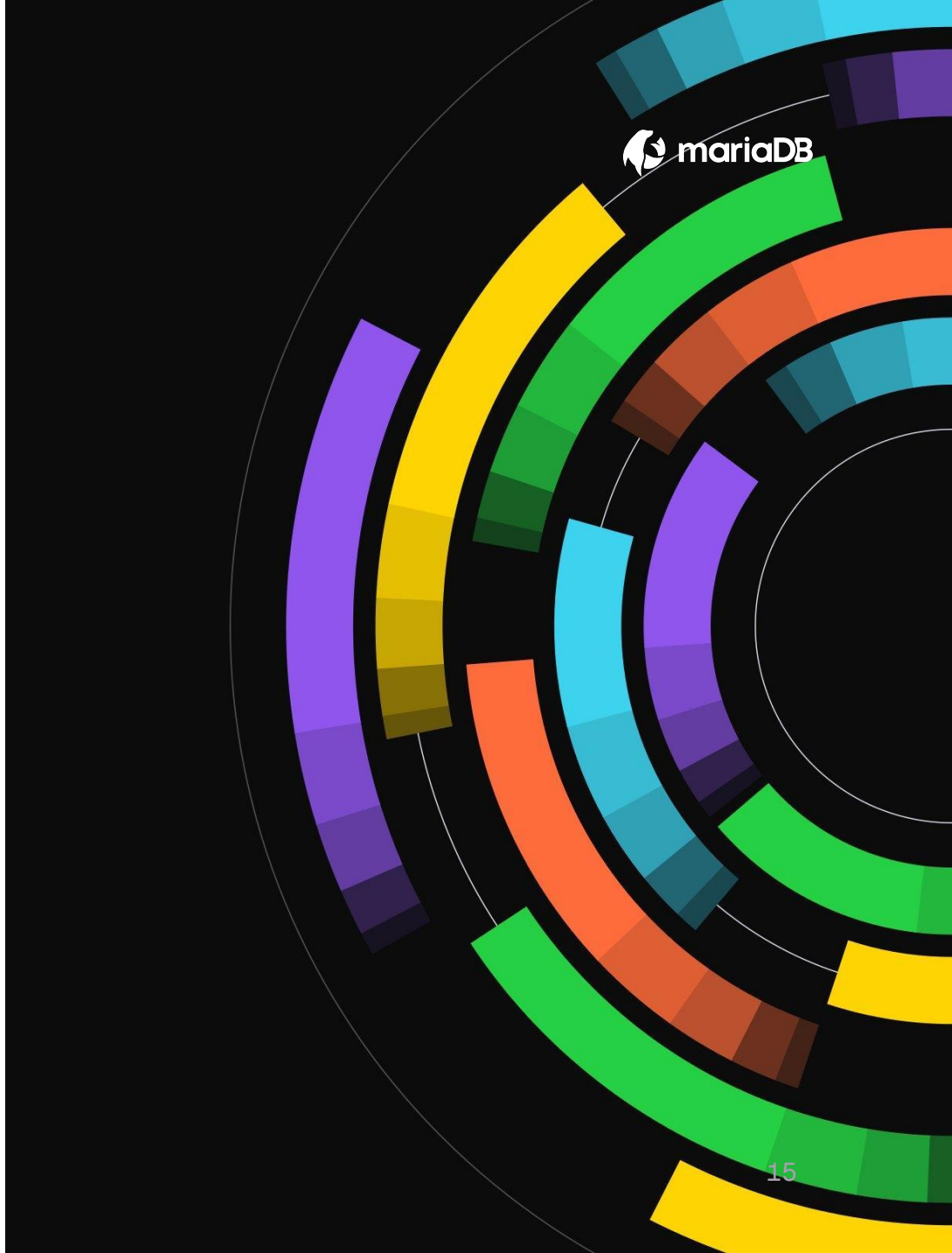
Why RAG helps

Why enterprise AI needs databases

Enterprises want to adopt AI using their own data. Current GenAI models can hallucinate without proper context

Customers need to **ground** AI in their **private databases**

RAG solves this, setup can be complex



Why RAG

Retrieval-augmented generation

- RAG creates more accurate, fact-based GenAI answers based on data of your own choice, such as your own manuals, articles or other text corpora. RAG answers are more accurate and fact-based than general Large Language Models (LLM) without having to train or fine-tune a model.
- RAG (Retrieval-Augmented Generation) applications scale with maximum efficiency and zero latency penalty.
- Keep an eye out here, we have more coming that takes RAG creation from 6 steps to 2 steps in the cloud

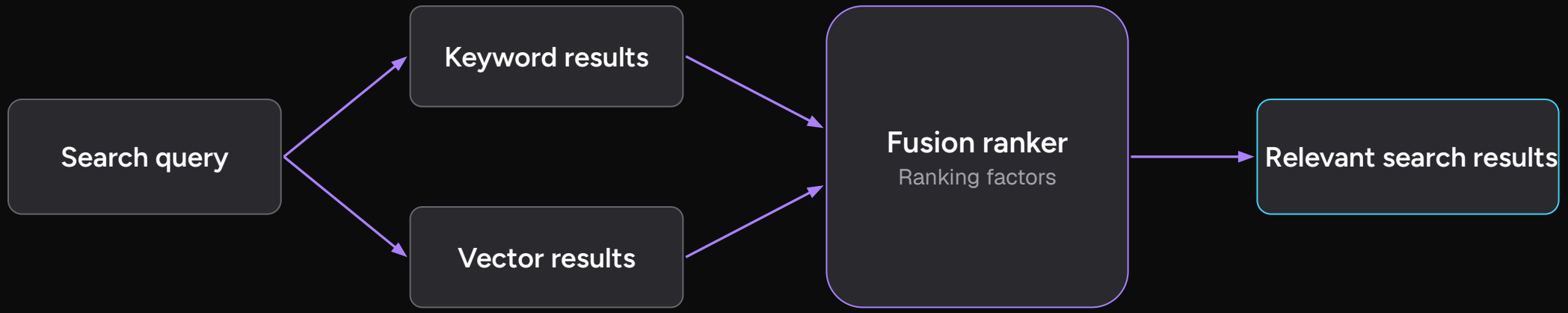
Unstructured vs structured RAG

Dimension	Structured RAG	Unstructured RAG
Data source	Relational databases (SQL, Snowflake), CSVs, Excel, or tables with a fixed schema.	PDFs, Word docs, emails, transcripts, and free-form prose.
Retrieval mechanism	Text-to-SQL or schema-driven API queries (translating natural language into database commands).	Semantic vector search (embedding text chunks and searching by conceptual similarity).
Precision and determinism	High determinism; exact counts, aggregations, and filtered values (e.g., “What was Q1 revenue?”).	Broad conceptual coverage; nuance, explanations, and multi-document synthesis (e.g., “What do our policies say about remote work?”).
Main failure mode	Mistranslating natural language into a bad SQL query or missing schema definitions.	Irrelevant vector matches, poor chunking boundaries, or missing specific facts in long passages.

Differ primarily in the format of the data they target, how they retrieve information and what type of questions they answer best

The value of hybrid search

Hybrid search unifies keyword and vector retrieval to provide production-grade context accuracy



Exact precision

Retrieves literal matches for SKUs, part numbers, and technical terms missed by semantic models.

+

Semantic depth

Understands intent and context, finding relevant concepts even without matching terminology.

→ Effectively handles queries that require both conceptual understanding and literal, exact term matching, boosting search accuracy

Retrieval engineered for accuracy

mariaDB

Unstructured RAG

Hybrid search

Combine semantic meaning with exact-term matching so nothing relevant is missed

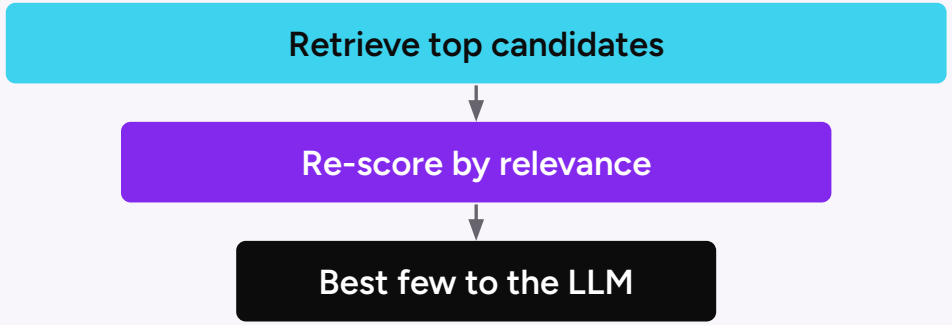
1 **Semantic recall**
Understands intent and context, even without matching keywords.

2 **Lexical precision**
Catches literal matches, including SKUs, part numbers, codes, and identifiers.

Fused with Reciprocal Rank Fusion (RRF)

Cross-encoder reranking

A second pass re-scores top candidates so only the most relevant context reaches the model, reducing hallucinations

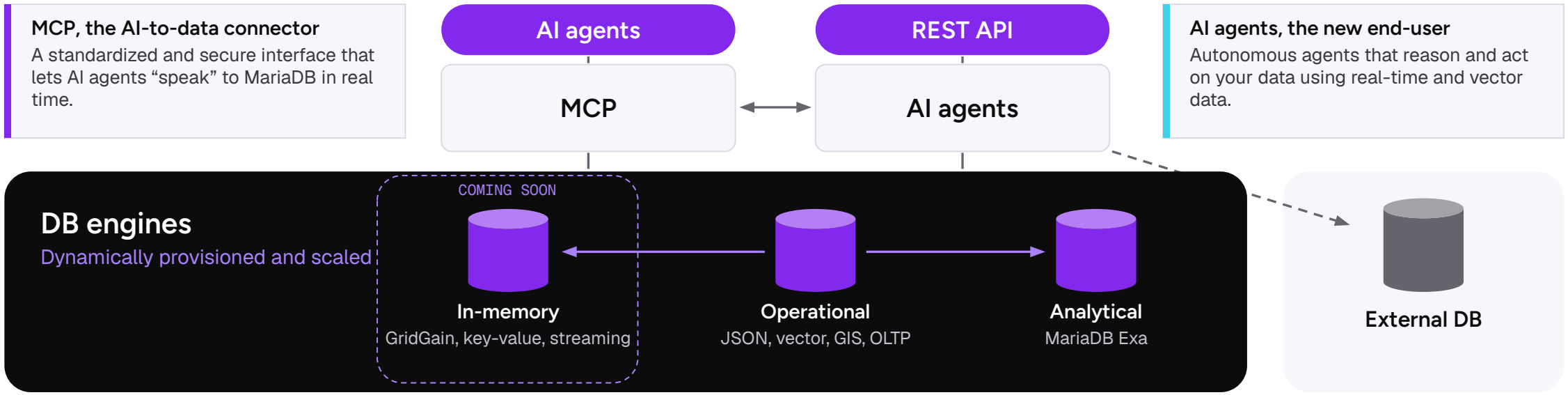


Backends FlashRank (built-in) · Cohere (external)

Search modes Semantic · Full-text · Hybrid (RRF)

A converged architecture for the agent-ready cloud

MCP, the AI-to-data connector
A standardized and secure interface that lets AI agents “speak” to MariaDB in real time.



AI agents, the new end-user
Autonomous agents that reason and act on your data using real-time and vector data.

Control plane and data plane architecture

Fully powered by K8s

Multicloud and BYOC

Azure aws Google Cloud

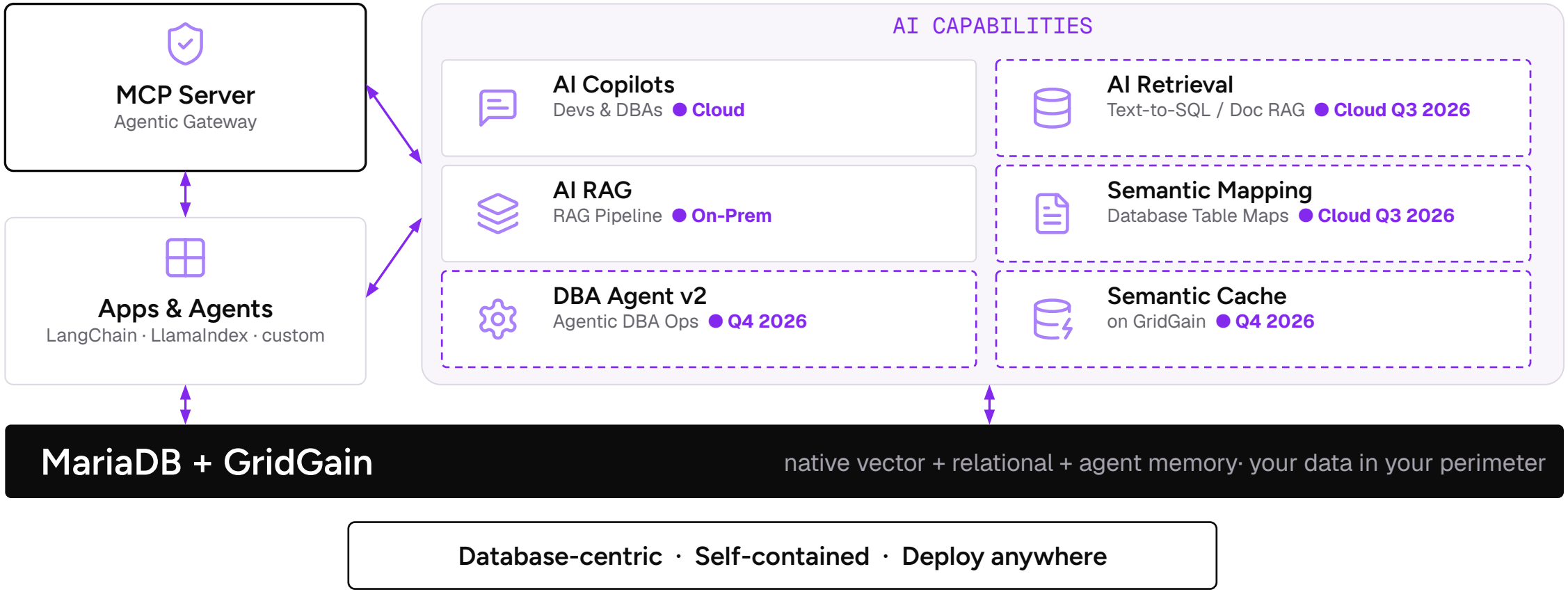
05

05 / The MariaDB AI platform

MariaDB

The MariaDB AI stack

USE CASE Build AI on your own data — governed, multi-cloud and on-prem.



Database-centric · Self-contained · Deploy anywhere

Agent memory, solving context drift and memory loss

Multi-step financial audit and compliance agent, 50+ tool calls and multi-agent handoffs across SEC filings

WITHOUT AGENT MEMORY

Context drift and bloat

Example, uncached 50-step audit workflow (isolated memory)



1. Fresh context
Audit starts clean; parses initial 10-K report.

2. Bloat and dump
Context fills with 100s of raw filings; agent dumps state on handoff.

3. Failure and costs
Drops audit subgoals, repeats tool calls and incurs maximum token bills.

WITH MARIADB AGENT MEMORY

Checkpointed memory

Example, checkpointed audit execution (shared agent state)



Step checkpoints
Auto-saves audit progress at targeted steps; instant recovery if step or workflow fails.

In-memory store
Stores extracted financial ratios in GridGain; zero context bloat.

Clean handoffs
Passes verified audit state to next AI agent with zero re-computation.

ARCHITECTURE AND VALUE DELIVERED BY MARIADB GRIDGAIN

1 AGENT MEMORY

1. Long-running sessions
Saves state at every step with expiry settings. Prevents context drift and lost subgoals.

2. Multi-agent handoffs
Challenge is work duplication between agents. The solution is shared namespaces across workers.

2 IN-MEMORY STORE

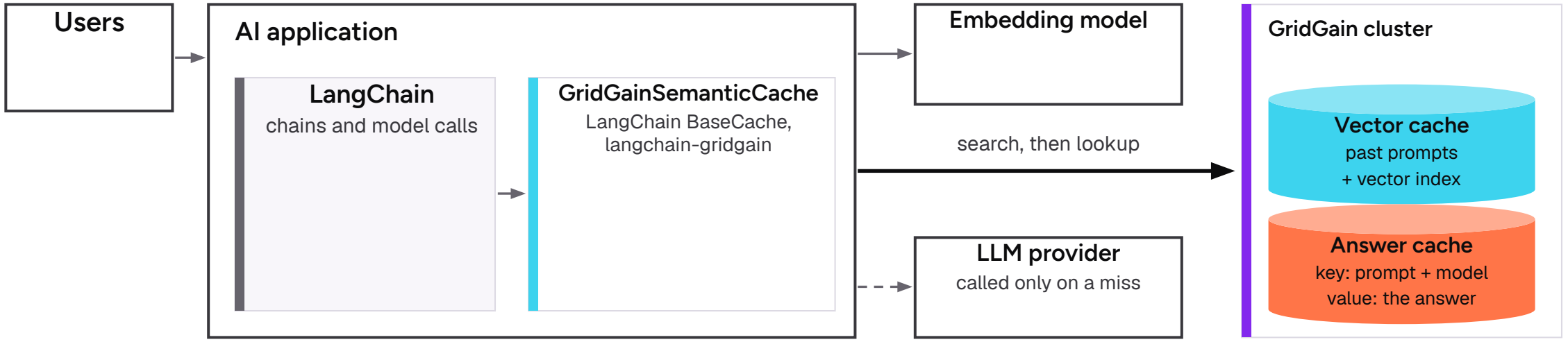
Stays fast as memory grows
Durable key-value and vector storage eliminates context window bloat and keeps execution latency sub-second.

3 BUSINESS VALUE

Lower costs and high efficiency
Prevents duplicate model processing and redundant steps. Dramatically cuts token expenditure in high-concurrency runs.

Semantic cache for sub-second AI response and lower cost

Semantic cache architectural pattern and execution flow



Primary use cases

High-concurrency AI chatbots, customer support assistants, and enterprise search platforms where recurring or semantically identical queries waste compute and inflate token costs.

Enterprise strategic impact

Dramatically cuts operational LLM expenditure while enabling predictable sub-second response times across high-concurrency production deployments.

What a semantic cache saves

Cache miss versus cache hit, with third-party benchmark results

1 Cache miss (full LLM call)

HIGH COST

~2,500ms

latency

$300.00

10k queries at $0.03 / query

FAQ example “How do I cancel my plan and get a refund?”

2 Cache hit (semantic match)

99% SAVINGS

<10ms

latency

$1.00

10k queries at $0.0001, saves $299

FAQ example “What’s the process to request a refund and cancel my plan?”

THIRD-PARTY BENCHMARK, AWS STUDY, 63,796 REAL CHATBOT QUERIES

Up to

86%

cost reduction

Eliminates redundant model inference costs by returning cached embeddings and responses directly.

Up to

88%

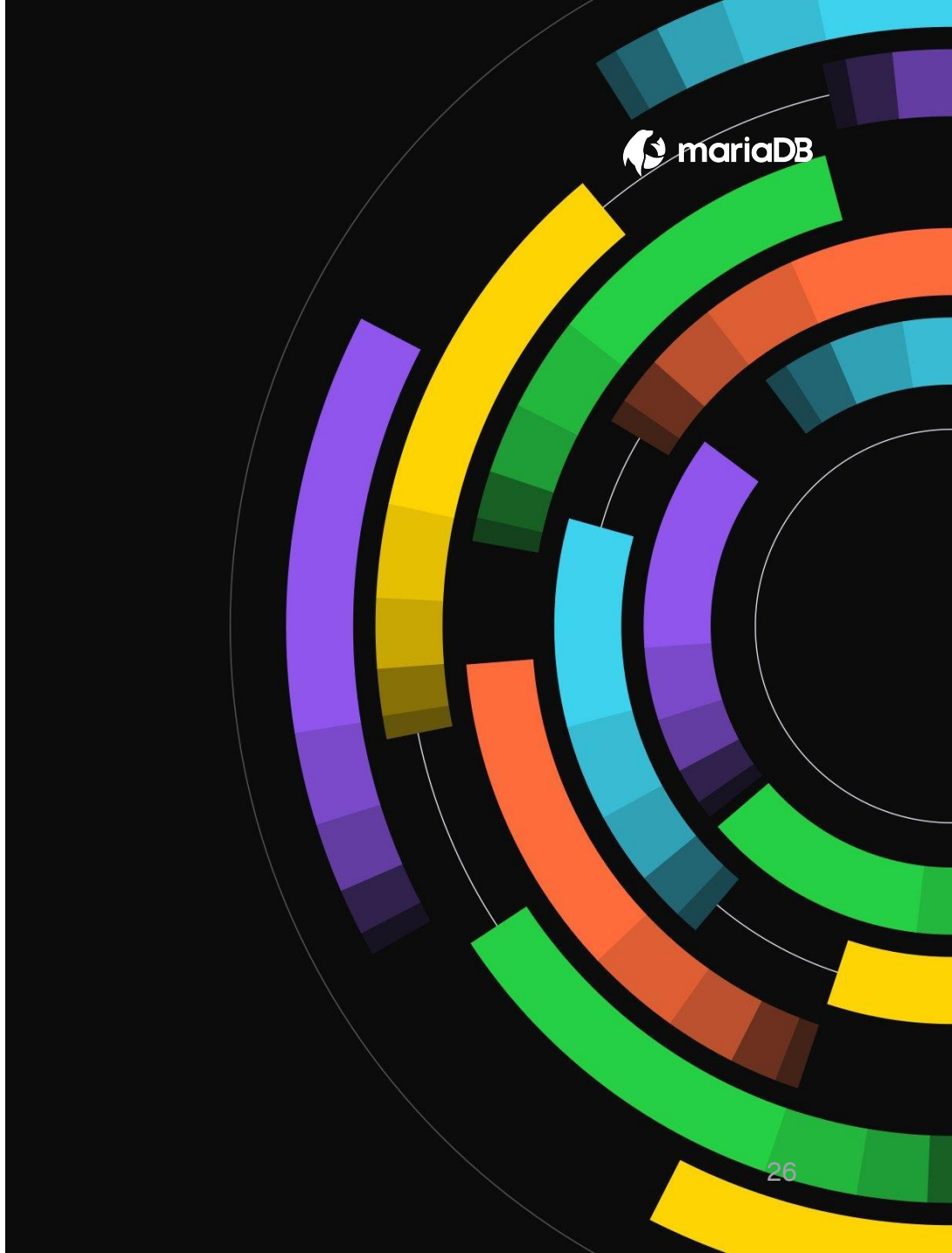
latency cut

Replaces multi-second LLM API calls with sub-millisecond in-memory vector lookup in GridGain cluster.

A converged, AI-ready operating platform

MariaDB + GridGain fuses the trusted system of record with high-speed in-memory capabilities into a single converged platform

- Stop database sprawl: one platform for transactions, analytics, and AI search
- Instant speed: built-in caching delivers sub-millisecond results for AI agents
- Trusted foundation: enterprise-grade reliability extending our MariaDB proven track record



Community vs Enterprise Platform

Match the edition to how critical the application is

Community Server

- For rapid innovation and new feature testing
- A community-driven open source project
- Focused on innovation over security, stability and long-term support

Enterprise Platform

- For mission-critical applications
- Long-term stability, security and support
- MaxScale for high availability
- Using Community Server for critical needs carries significant risk and cost

06 / Wrap up

Conclusion

Conclusions

AI is only just getting started, so don’t be afraid to play

- Look for ease of use
- Look for trustworthiness, with agent layers where agents check agents
- Limit the Frankenstein stack
- Keep security and performance top of mind

Use to provide feedback



Look to AI for your Financial Reporting Needs in an AI World

Learn more...

- Continue the conversation

Booth T10 Hall 2

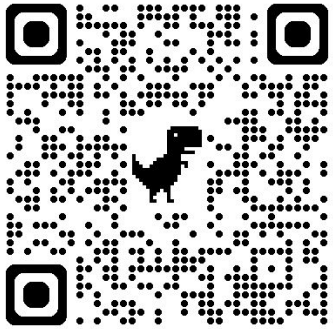
- Visit our new website

mariaDB.com

- Don't miss our upcoming webinar

Built for the Agentic Era: Introducing the MariaDB Enterprise Platform 2027

Scan to register →



mariaDB

Thank you

Kathy Gibbs

kathy.gibbs@mariadb.com

Follow me on Linkedin →

