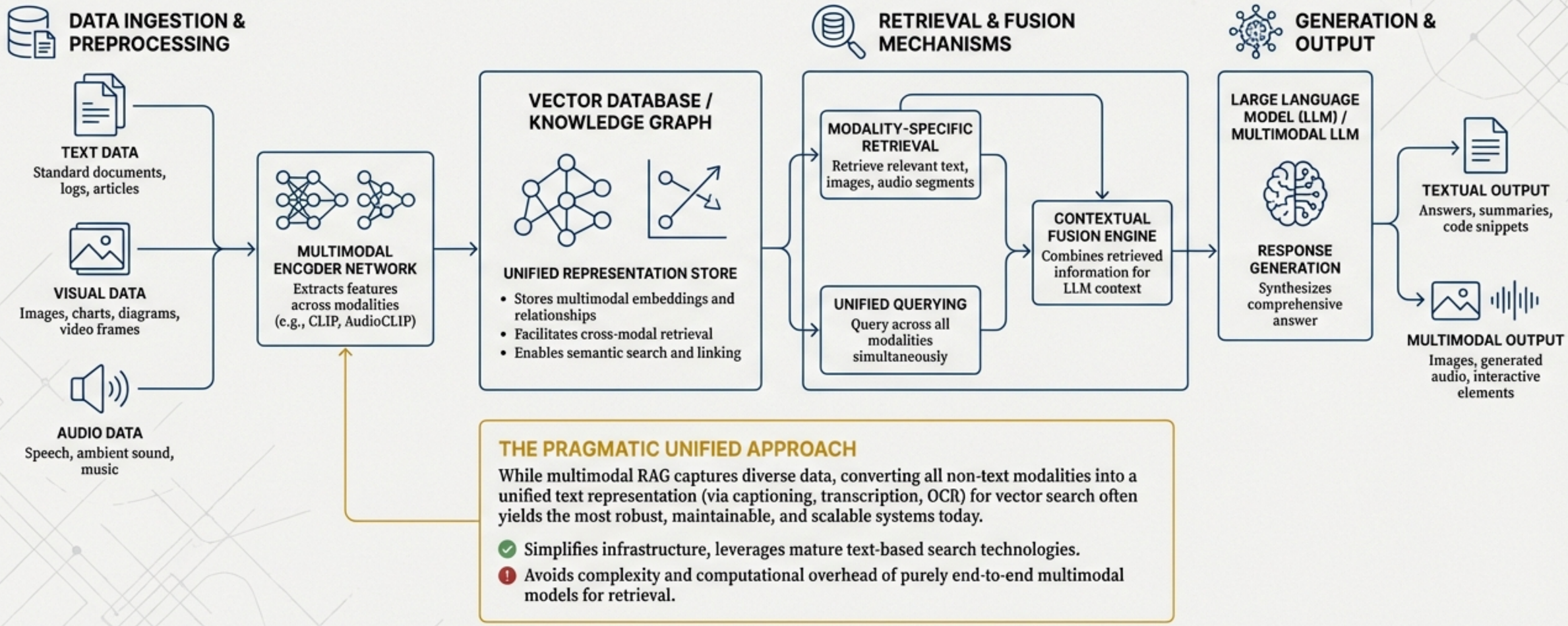
Multimodal RAG: The Complete Architectural Landscape

And why a unified, text-based approach is the most pragmatic choice for most systems.



THE PRAGMATIC UNIFIED APPROACH

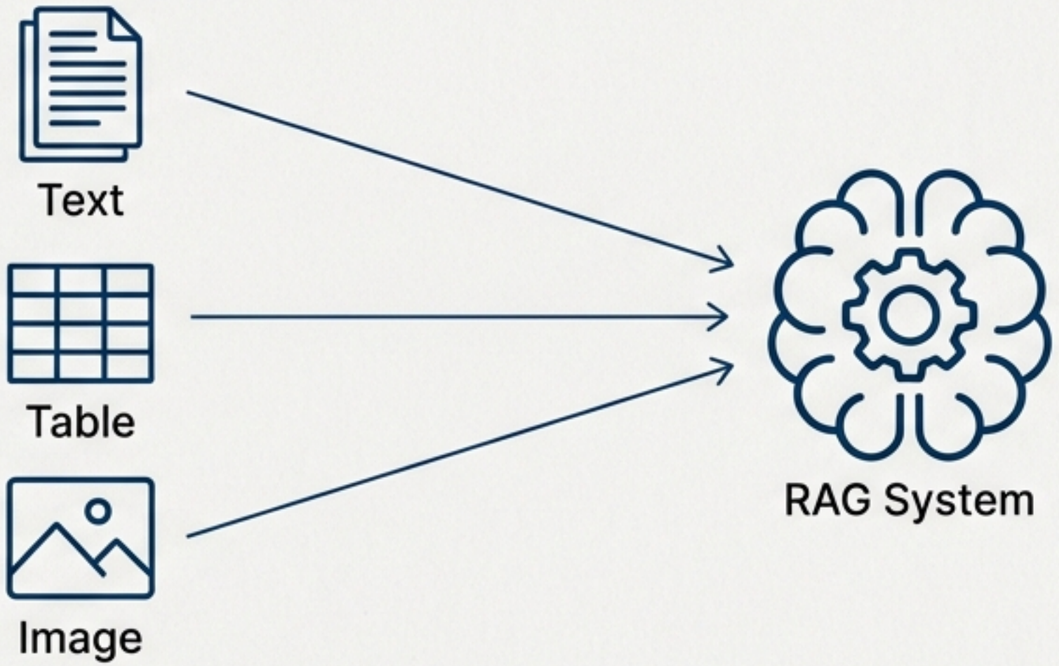
While multimodal RAG captures diverse data, converting all non-text modalities into a unified text representation (via captioning, transcription, OCR) for vector search often yields the most robust, maintainable, and scalable systems today.

- ✓ Simplifies infrastructure, leverages mature text-based search technologies.
- ! Avoids complexity and computational overhead of purely end-to-end multimodal models for retrieval.

The Core Challenge: Making Sense of Mixed Data

A Multimodal RAG system must process and retrieve information from a mix of unstructured data:

- Text Documents: Reports, articles, internal wikis.
- Tables: Structured data within documents.
- Images: Charts, diagrams, application screenshots.



How should images be stored and retrieved in a RAG system?

There is No Single “Correct” Answer, Only Trade-offs

Each architectural approach optimizes for a different set of priorities.
The “best” choice depends entirely on your specific constraints.

Accuracy

How precise are the results?

Cost

What is the price per query and for storage?

Simplicity

How easy is it to build, deploy, and maintain?

Scalability

Can it handle high query volume efficiently?

Engineering Effort

How much developer time is required?

Path 1: The Specialist — Separate Collections per Modality

How It Works:

Text and tables are processed by a text embedding model.
Images are processed by a separate, specialized multimodal embedding model.
This creates two distinct vector collections.
Queries are routed to the appropriate collection.

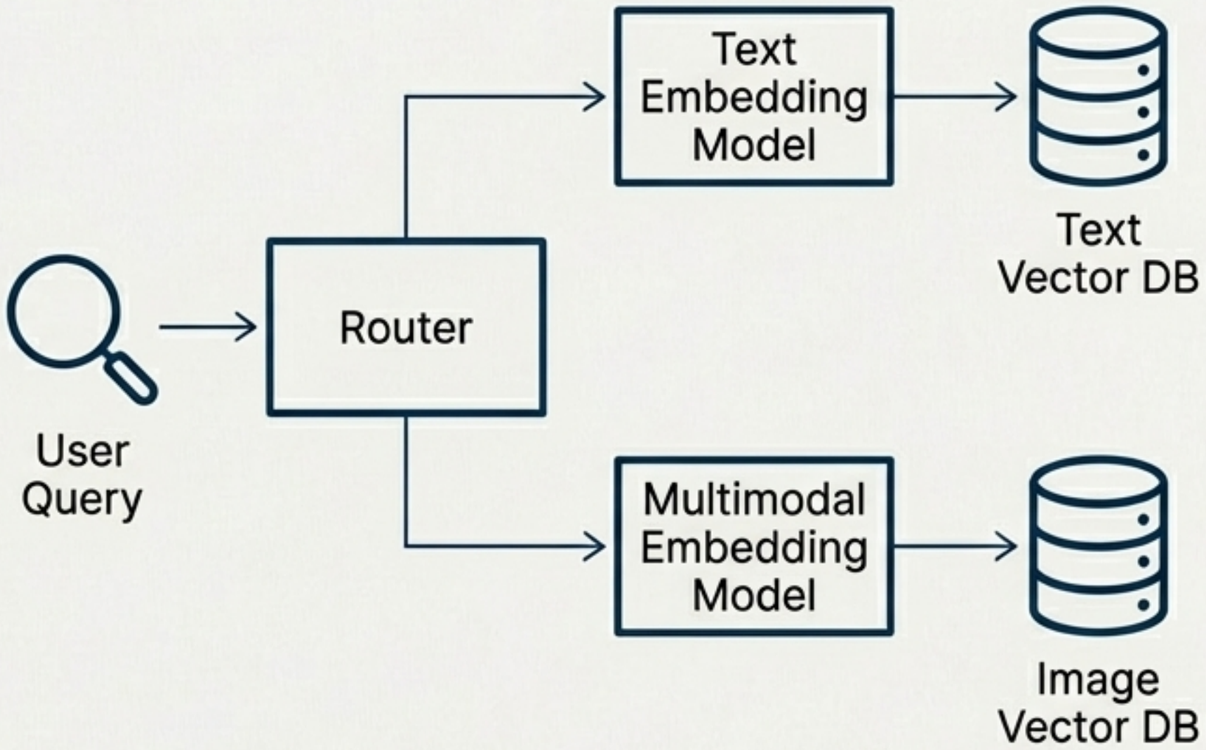
Key Trade-offs:

✅ Pros: Best-in-class text retrieval quality; enables true image similarity search; clean separation of concerns.

❌ Cons: Requires multiple models and collections; introduces credential complexity (e.g., Vertex AI); often sends the image to the LLM again at query time, increasing cost and latency.

Best For:

Vision-heavy systems where image similarity is the primary goal and query frequency is low.



Path 2: The Unifier — A Single Collection with Multimodal Embeddings

How It Works:

A single, powerful multimodal embedding model is used for everything. Text, tables, and images are all embedded into the same vector space and stored in one collection.

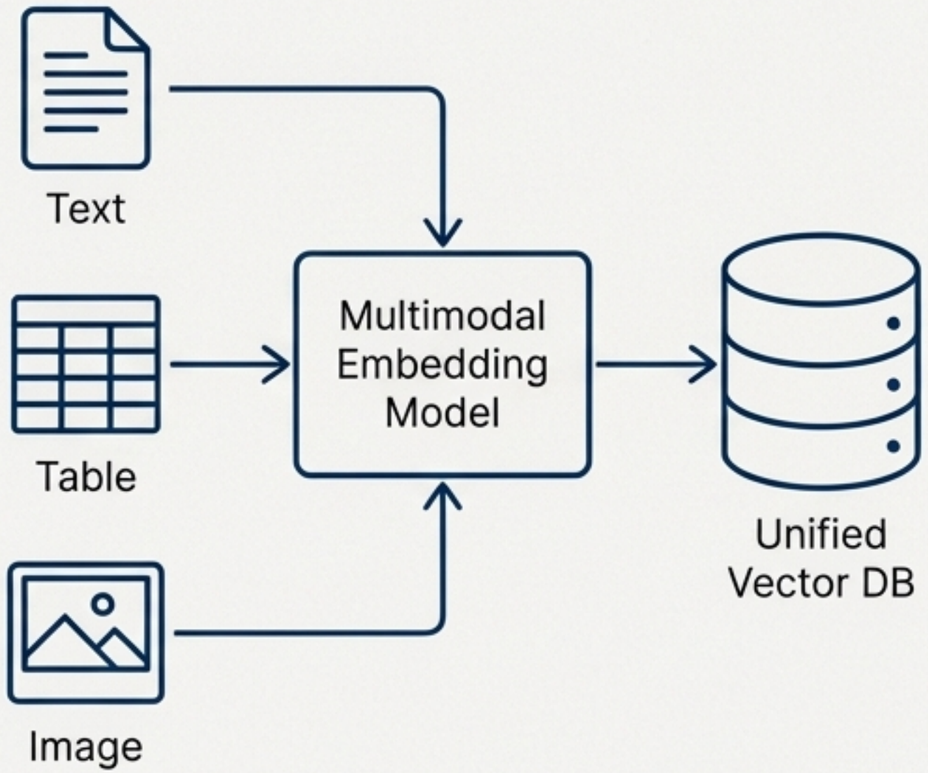
Key Trade-offs:

✅ **Pros:**
Conceptually clean with a single collection; allows for true multimodal similarity search (e.g., text finds similar images).

❌ **Cons:**
Sacrifices hybrid/sparse search capabilities, weakening keyword recall for text; multimodal embeddings are expensive; suffers from API limitations and is harder to tune for text-heavy RAG.

Best For:

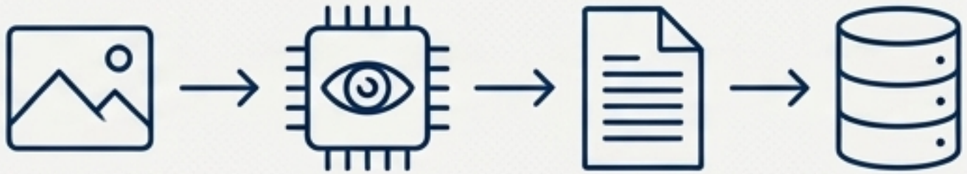
Research prototypes and image-first applications where visual reasoning is paramount.



Path 3: The Pragmatist — Images as Text Descriptions

How It Works:

Images are pre-processed through a vision model (like an LLM) to generate rich, text-based descriptions. These text descriptions are then embedded and stored just like any other piece of text. The original images are not indexed.



Key Trade-offs:

✅ Pros: Uses a single text embedding model and a single collection; zero image inference cost at query time; predictable, low-cost operations; simple to deploy and scale.

❌ Cons: Loses pixel-level image similarity search; retrieval quality is entirely dependent on the quality of the generated text descriptions.

Best For:

Document-heavy RAG with informational images (charts, diagrams, screenshots) and cost-sensitive systems.

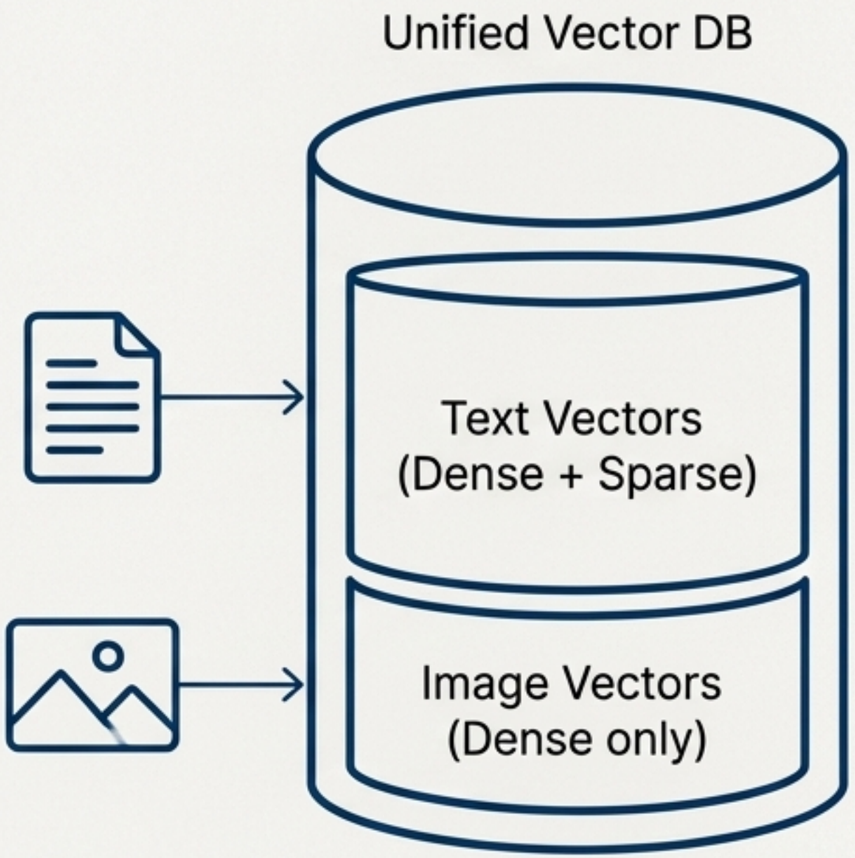
Path 4: The Power User — Hybrid Named Vectors

How It Works: A single collection is used, but it stores multiple, named vector types for each document. Text chunks get text embeddings (dense + sparse). Images get multimodal embeddings (dense only). The vector database handles the complex internal routing during a query.

Key Trade-offs:

- ✅ Pros: Achieves the best of both worlds: hybrid search for text and true similarity for images, all in one collection.
- ❌ Cons: Very complex to set up and manage; requires advanced query logic and dealing with multiple vector dimensions; difficult to debug and teach to new team members.

Best For: Large-scale enterprise platforms with dedicated infrastructure teams and long-term multimodal product goals.



Path 5: The Maximalist — Multimodal at Retrieval AND Reasoning

How It Works:

The system retrieves the actual source images. These images are then sent to a multimodal LLM *for every single query* to perform real-time visual reasoning.

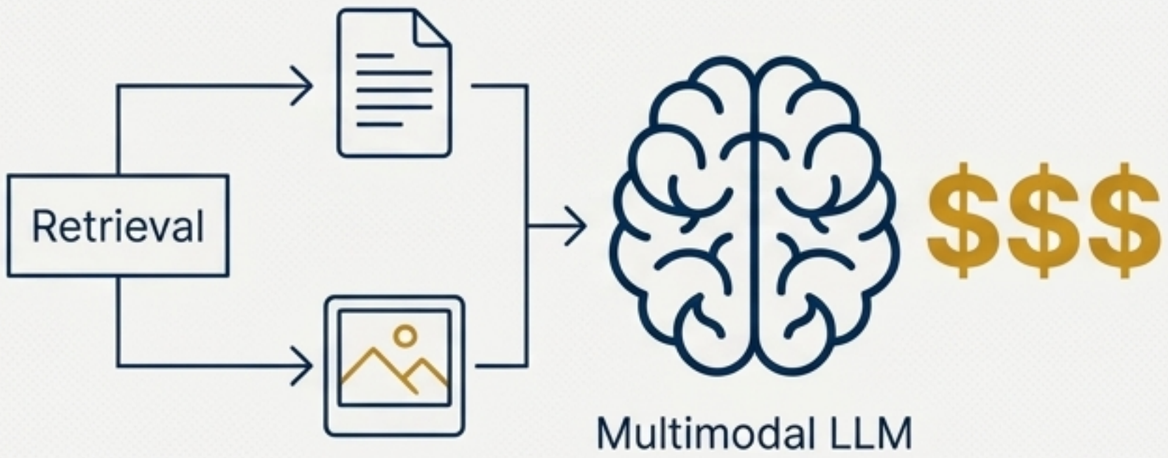
Key Trade-offs:

✅ **Pros:** Highest potential for deep visual reasoning; no information is lost from the original image.

❌ **Cons:** Extremely high cost and high latency per query; scales very poorly; not suitable for applications with frequent RAG calls.

Best For:

Low-frequency, expert systems like medical image analysis or specialized design review tools.



The Decision Matrix: All Approaches Side-by-Side

Approach	Collections	Embeddings	Image Cost at Query	Complexity
1. Separate Collections	Multiple	Multiple	High	Medium
2. Unified Multimodal	One	Multimodal	Medium	Medium
3. Image → Text (Chosen)	**One**	**Text only**	**Zero**	**Low**
4. Named Vectors	One	Multiple	Low	High
5. Full Multimodal Reasoning	Multiple	Multimodal	**Very High**	High

Our Decision Was Guided by a Clear Set of Constraints

For our use case, the following operational requirements were non-negotiable:

⚡ **High Frequency:** The RAG system will be called frequently by users.

💸 **Predictable Cost:** The cost per query must be low and predictable.

🚢 **Simple Deployment:** The architecture must be easy to deploy and maintain.

📊 **Informational Images:** Images are primarily charts, diagrams, and screenshots, not artistic photos.

📝 **Text is Primary:** The quality and recall of text-based search is the highest priority.

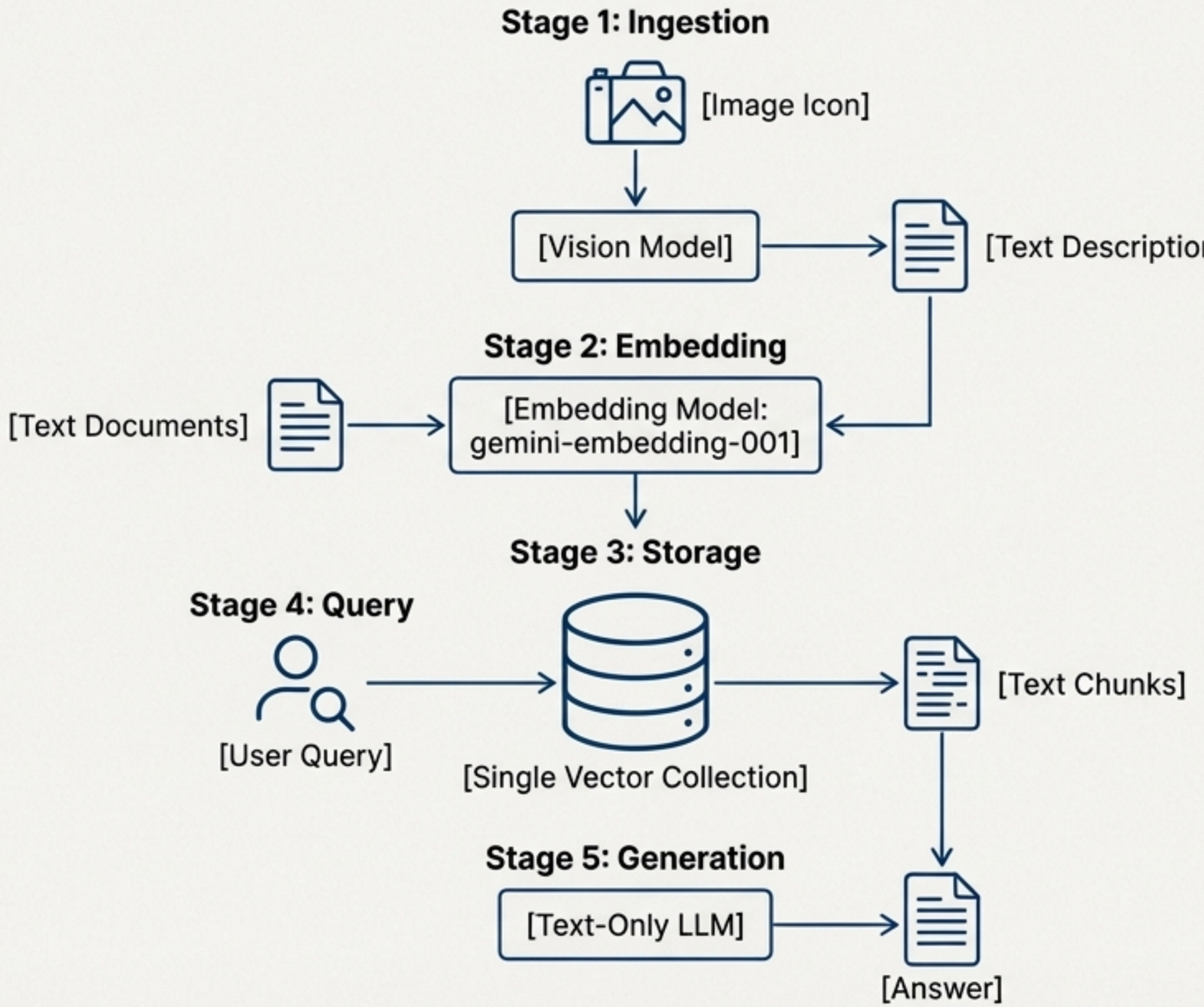
The Inevitable Choice: Convert Images to Text Once

> “Our decision is to convert images to text during ingestion, then treat everything as a unified text collection.”

This architecture deliberately avoids:

- The need for separate Vertex AI credential management.
- The high cost of repeated image inference at query time.
- The complexity of managing multiple, distinct vector spaces.
- The need for complicated query routing logic.
- The need for complicated query routing logic.

Our Final, Pragmatic Architecture



The Key Takeaway

“Good system design is about trade-offs, not features.”

We are not avoiding multimodality. We are **strategically controlling where** multimodality is used—at ingestion time, where the cost is paid only once.

Multimodal RAG does not mean multimodal inference everywhere.

The smartest systems are:

Cheap
Predictable
Simple
Scalable