Whitepaper

Why RAG Systems Fail

How to Fix Them with the Aspect Database

Aspected

Executive Summary

Vector search made it possible to retrieve documents by meaning rather than keywords, but most systems still embed only unstructured text and treat metadata as external filters. Historically, metadata was introduced to compensate for the limits of keyword search, yet it never truly became part of relevance itself. With vector search, the same happened again. Aspected bridges this gap by transforming meaningful document properties into vectors and combining them with semantic embeddings into a single representation. The result is a unified, weighted, multi-dimensional search that runs in one query, expresses relevance more naturally, and scales efficiently.

The Problem: Why Retrieval Breaks in Enterprise AI

The evolution of search

As digital information exploded in volume and diversity, search systems had to evolve quickly. Early information retrieval relied on lexical techniques: inverted indexes, keyword matching, and boolean logic. These approaches worked well for a long time, but struggled with ambiguity, synonymy, context, and human error. This limitation becomes particularly visible in modern RAG systems, where retrieval quality directly determines generation quality. More broadly, many failures in enterprise AI systems are not caused by the reasoning capabilities of large language models, but by the retrieval layer selecting the wrong knowledge in the first place. In an attempt to fix these problems, reliance on metadata started to grow.

With the adoption of content management systems, structured fields like content type, author, department, creation date, category, and sensitivity labels became common. This metadata allowed systems to constrain search space and restore precision where keywords failed, and the reliance on metadata for accurate search started to grow. For a long time, this worked well enough: keyword search for recall, metadata filters for control.

However, this approach still treats metadata as adjacent to search, not as a core part of it. As datasets expanded and started to include a wider variety of content types, such as documents, images, audio, and semi-structured content, limitations of this approach became clear. Metadata helped but required careful curation, strict schemas, and constant governance. It improved precision but struggled with properly encapsulating meaning. Then vector search arrived.

Meaning as geometry: The rise of vector databases

Vector search changed the game by allowing systems to retrieve documents based on semantic similarity rather than exact matches. Documents, queries, and other entities are encoded as numerical vectors in high-dimensional space, where proximity reflects meaning, rather than a high overlap in textual content.

With advances in large language models, embeddings became good enough to serve as a foundation for semantic search, recommendations, question answering, and retrieval-augmented generation (RAG).

Suddenly, search systems could answer questions like “find documents similar to this idea” rather than “find documents containing these words.” This was a major leap forward, but it also quietly repeated an old pattern.

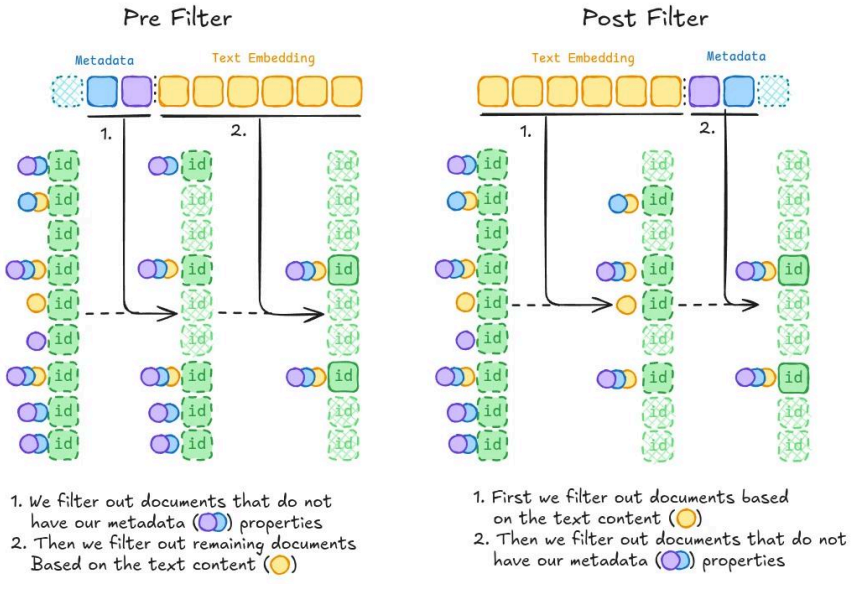
Similarity is not the same as correctness: a document can be semantically close to a query while still being outdated, unapproved, or irrelevant in a given context. Vector databases are highly effective at optimizing for semantic similarity, whereas enterprise AI systems require context-aware relevance.

Despite their power, vector embeddings typically can only work with unstructured content, usually text. Structured metadata such as timestamps, file types, ownership, sensitivity labels, and source systems are once again pushed aside, causing problems with ambiguity and context. In short, similarity is not enough. Retrieval needs context.

Why Current RAG Pipelines Keep Getting More Complex

Fixing (vector) search with metadata (again)

A somewhat common but naive approach to incorporating metadata in vector search is appending it as plain text to the unstructured content before embedding. However, this method lacks control over the importance of each structured property and often results in "muddy vectors," where the metadata is drowned out by the unstructured content itself.
This problem can be understood as a form of metadata dilution. In practice, a small number of tokens encoding structured constraints such as dates, classifications, or identifiers are combined with much larger amounts of unstructured content. Once compressed into a single embedding, the semantic signal of the content can dominate, making the metadata harder to preserve as a meaningful retrieval signal.
A more refined approach is to apply metadata-based filters on the index, either before (pre-filtering) or after (post-filtering) performing the actual vector search.
This approach is also closely related to hybrid search systems, where vector similarity is combined with keyword search or structured filters, often resulting in multi-stage pipelines. While hybrid search improves many production systems, it still usually combines separate retrieval signals at query time rather than representing them within a unified retrieval model.



With pre-filtering, structured constraints are applied to the index prior to running the vector search, excluding vectors that do not match the filters. While straightforward, this acts as a hard gate and can eliminate semantically relevant results that fall just outside the filter criteria. In addition, pre-filtering prevents the use of common vector search optimizations such as Hierarchical Navigable Small World (HNSW) graphs, which require access to the full index. As a result, searches often degrade to brute-force comparisons, which is only feasible when the filters are restrictive enough to keep the search space small. With broader or more general filters, large portions of the index must still be searched, causing performance to degrade to the point where real-time use becomes infeasible.

Post-filtering takes the opposite approach: the vector search is performed over the full index, and structured filters are applied only to the retrieved results. This ensures that all items are considered semantically and allows the use of optimizations like HNSW, since the search space remains fixed and optimizable. However, because filtering happens only after retrieval, the final result count becomes non-deterministic which means potentially significantly less documents are returned that initially requested. To compensate, systems typically over-fetch results during the vector search to ensure enough items remain after filtering. Even then, an important failure mode remains: the correct document may never enter the initial Top-K candidate set, meaning it is excluded before filtering or reranking can help. This overshooting is inefficient, can significantly impact performance, and still risks missing important results, especially when filters are highly restrictive or the queried topic is semantically dominant across the index.

In many modern systems, this is further complemented with reranking models, often based on cross-encoders or LLMs, to refine final results. As a result, many teams end up assembling increasingly complex retrieval stacks, adding filters, rerankers, and sometimes graph-based layers to compensate for missing context. With both pre- and post-filtering, metadata remains a corrective mechanism layered on top of search rather than part of the retrieval model itself. This architectural separation introduces a set of familiar problems:

- **Metadata acts as a hard constraint**, not a graded signal
- **Important context (time, type, classification) cannot smoothly influence relevance**
- **Multiple query stages** increase latency and operational complexity.

We have effectively fallen into the same trap as before: metadata is essential, yet it remains external to the search itself. Instead of becoming part of the solution, metadata is still being treated as a fix. As a result, existing solutions are not always ideal, introducing inefficiencies that make it hard to scale and are not resilient to imperfect metadata or human error. In practice, this often leads to silent retrieval failures. The system does not crash or return an obvious error; instead, it retrieves a plausible but contextually wrong document with a high confidence score. In RAG systems and AI agents, these failures propagate downstream, where they may appear as generation issues even though the root cause lies in retrieval. To mitigate these problems, we developed Aspected, built around an **Aspect Database** that brings context directly into retrieval.

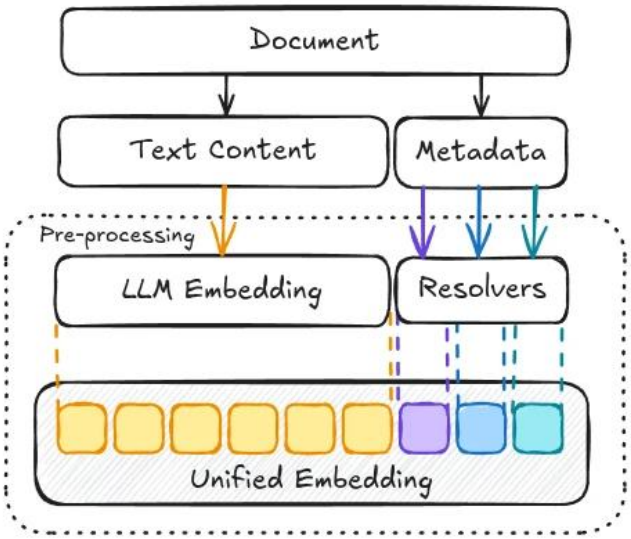
The Architectural Shift: Introducing Aspected

Aspected: unifying meaning and structure

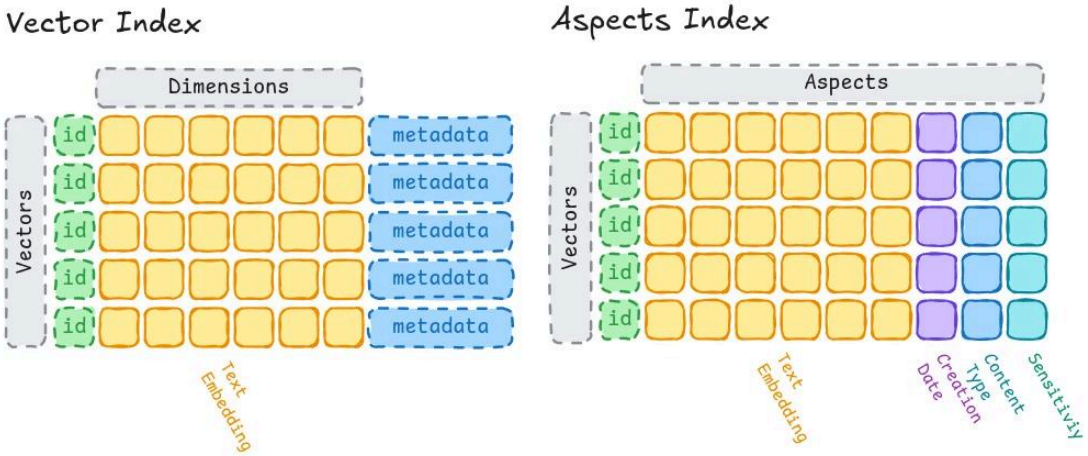
The core idea behind Aspected is straightforward: rather than forcing a choice between semantic embeddings and structured metadata, all meaningful properties of a document are treated as first-class components of the search vector. This concept emerged from our work at Xillio, a data migration company with more than two decades of experience handling large volumes of both structured and unstructured data. As organizations increasingly began using their data for RAG systems and AI agents, a consistent pattern emerged: metadata was rarely missing or inadequate; the real issue was that it was treated as secondary to semantic content.

This shift also changes the fundamental unit of retrieval. Instead of treating documents or text chunks as the primary unit, Aspected models knowledge as structured entities composed of multiple aspects. These can be understood as knowledge objects, where meaning, context, and constraints are represented together as a coherent whole. These aspects can include semantic meaning, intent, workflow position, role relevance, authority or validity, and constraints or risk conditions.

In Aspected, these document properties are simply called aspects. They encompass everything that provides meaningful context about a document, such as its content type, when it was created or last modified, its sensitivity or classification level, where it is stored or which system it originated from, and even geographic information like coordinates or postal codes.



A simplified overview of how a document can be prepared into a full unified embedding, in aspects search. Each aspect is encoded into its own vector representation, designed to reflect how that property behaves in the real world. Categorical attributes are projected onto continuous ranges, temporal values are normalized along a linear scale, and semantic content is represented using high-dimensional embeddings generated by models such as LLMs. The goal is not to flatten all information into a single homogeneous signal, but to preserve the distinct characteristics of each property while making them comparable within a shared vector space.



On the left: a traditional vector index, where metadata and embeddings are handled separately. On the right: the Aspect Database, where content and contextual aspects are combined within a unified retrieval representation. (Vector index depiction greatly inspired by the amazing team at QDrant)

These individual aspect vectors are then combined into a single, unified full-aspect embedding that represents the document as a whole. Unlike traditional vector databases, where individual dimensions are typically opaque and lack human-interpretable meaning, the full-aspect embedding is structurally meaningful. Each contiguous segment corresponds to a specific aspect of the document, making the vector not just a numerical representation, but a coherent, multidimensional description of the document itself.

Metadata as signal, not constraint

Historically, metadata has been used to compensate for the limitations of search technologies rather than to enhance them directly. In keyword-based systems, where queries lacked semantic understanding, metadata was introduced to improve precision and narrow results. In vector search, the situation reversed: meaning was captured effectively, but structure and control were missing, so metadata reappeared as a set of external constraints layered on top of the search.
Aspected represents a natural evolution beyond this trade-off. Instead of treating semantic meaning and metadata as fundamentally different mechanisms, both are modeled as signals of the same kind and can therefore contributing equally when searching. Instead of asking “should this metadata filter apply?”, the system asks “how much should this aspect matter right now?”. This shift transforms metadata from a rigid gatekeeper into an active participant in ranking, allowing relevance to emerge smoothly from multiple dimensions rather than being enforced through hard constraints, while maintaining the efficiency of single-step optimized vector search. This is particularly important in enterprise environments, where relevance is not only about meaning, but also about authority, validity, and the role or context of the user interacting with the system.

Why It Changes Search Behavior

This reflects a broader shift in AI system design: retrieval comes before reasoning. The quality of the knowledge selected directly determines the quality of the generated output.
By embedding metadata into the vector space, rather than filtering around it, Aspected enables a more expressive and efficient retrieval model:

- **Single-query, multi-dimensional search**
  Semantic similarity and structural context are evaluated together.
- **Weighted relevance instead of binary filters**
  Metadata can influence scoring proportionally, not just include/exclude results.
- **Aspect maskin**
  Queries can selectively ignore dimensions without biasing similarity math.
- **Cleaner system architecture**
  Fewer pipelines, fewer queries, less glue code.
- **Better performance at scale**
  One optimized ANN search instead of multiple sequential operations.

For example, instead of retrieving documents only by semantic similarity, a query could also incorporate the user's role, workflow stage, or validity requirements:

```
query = "Can I approve this insurance claim?"
role = "claims_manager"
workflow_stage = "approval"
```

In this setting, retrieval is shaped not only by topical similarity, but also by the context in which the question is being asked.

A further benefit is improved debuggability. When structured properties contribute explicitly to retrieval, engineers can reason more clearly about why a document was returned and which dimensions of relevance influenced the result. This makes retrieval behavior easier to inspect and refine than in pipelines where filters, embeddings, and reranking are layered separately.

In practice, this means search results feel more intent-aware and less brittle, especially in enterprise and knowledge-heavy environments. The broader vector search ecosystem is already slowly moving in this direction, where concepts such as hybrid representations are implemented. Aspected fits naturally into this evolution by unifying semantics and structure into a single representation. This way, vector search becomes not just a tool for semantic text retrieval, but a *general-purpose foundation for intelligent information access*.

This is particularly relevant in systems such as compliance assistants, enterprise copilots, customer service AI, and workflow-aware operational agents, where retrieving the correct information matters more than retrieving merely similar information.

System	Primary retrieval unit	Context handling	Output quality
Standard vector DB	text chunk	external filters / reranking	similar documents
Aspected	knowledge object	context-native retrieval	context-aware relevant knowledge

Conclusion

Vector search taught us how to make meaning computable, and metadata taught us how to impose structure and intent. Aspected is a natural extension of both, emerging when we stop treating them as competing approaches and instead allow them to work together.

For developers, this means fewer workarounds, clearer intent, and search systems that behave more like how humans actually think about documents. Documents are not just what they say, but what they are and why they matter.

Ultimately, retrieving the right information is not the same as retrieving similar information, and this distinction becomes critical as AI systems move into real-world, high-stakes environments.
We continue to develop and refine these ideas in collaboration with developers, partners, and enterprise teams exploring new retrieval architectures. Learn more at aspected.com