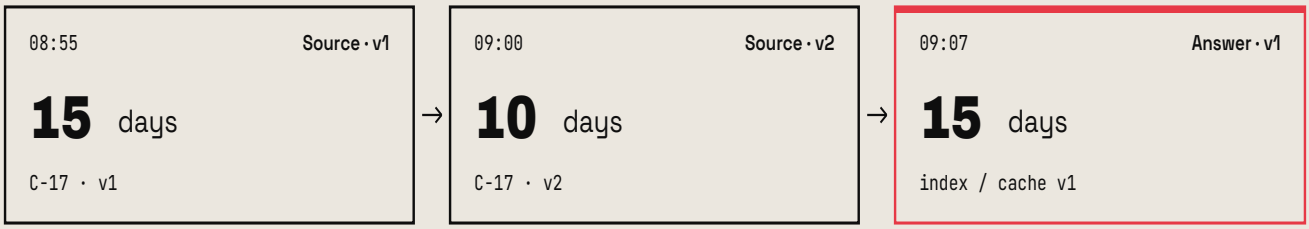
ENGINEERING NOTE · RAG

# Fresh database, stale AI: managing RAG freshness

An up-to-date source does not guarantee a current answer. This article follows the fictional document C-17 from source to answer and proposes four separate exercises: correction, deletion, revocation and processing recovery.

**C-17: trace the version**




Fictional example

Prepared 2 November 2026

Target canonical URL

https://datanalysis.re/ressources/artificial-intelligence/rag-freshness-source-index-cache

## Key takeaways

**01** Freshness must be checked across the whole chain: source, extraction, index, context and cache.

**02** MongoDB Search and Vector Search indexing is asynchronous; application caches follow their own rules.

**03** A useful citation identifies the version actually used, the passage and the access timestamp.

**04** Correction, deletion and revocation are distinct scenarios. A current answer can still be prohibited for a particular user.

**05** Recovery should preserve visible attempts while controlling their effects. Two attempts for one effect is a fictional expected result here, not a product guarantee.

THE COMPLETE ARTICLE

An answer can cite the right document while using an outdated version. In a retrieval-augmented generation system, or RAG, content is retrieved and then supplied to the model. Freshness depends on the entire processing chain. The source's modification date provides an initial reference. You also need to identify the indexed version, the context supplied to the model and any answer retained in a cache.

This article proposes a method for reviewing that chain. The example is fictional: it defines expected checks without describing a customer incident or a production test.

## Start with a verifiable timeline

At 08:55, version v1 of document C-17 states a processing time of fifteen days. At 09:00, a correction publishes version v2: the applicable processing time is ten days. At 09:07, an answer still says fifteen days and cites C-17.

In our scenario, the index and cache being used still rely on v1. The document reference is recognisable, but the answer no longer reflects the version currently in force.

Before changing the prompt, establish the path the answer actually followed. Which extraction populated the index? Which version was retrieved? Was the model called, or was an earlier answer reused? Without these traces, several possible causes remain indistinguishable.

## Map the copies that contribute to the answer

We propose a simple inventory: the business source, extracted text, chunks, embeddings, search index, assembled context, answer cache and history displayed to the user. Each item should have an identity, a known version and someone responsible for keeping it current.

This map avoids treating “the database” as a single object. A corrected source can coexist with an old chunk. A new index can coexist with a cache that is still valid under its own expiration rule. A history view can display an old answer as an archive without making it a valid answer to a current question.

For each copy, ask a concrete question: which event replaces it, invalidates it or prevents it from being read?

## Define freshness for the intended use

MongoDB documents asynchronous indexing and eventual consistency for MongoDB Search and Vector Search. Writing to a collection therefore does not imply that the change is immediately available to a search query. 1 2

This property belongs in the service design. An internal documentation library and an urgent operational instruction may tolerate different delays. For each use, define the acceptable lag and the expected behaviour when that limit is exceeded.

We recommend retaining several timestamps: source modification, change pickup, update of the searchable representation and answer generation. A single measurement hides which stage is running behind. If the required freshness cannot be established, the application can decline to reach a conclusion, request verification or clearly state the limitation.

Also verify how embeddings are managed. When the application generates them, updating them is part of its pipeline. A text modification alone does not guarantee that the embeddings have been recalculated.

## Give cached answers conditions for reuse

A cache reduces repeated work, but reuse should depend on context. For C-17, using only the question as a key would be insufficient: the same question may require a different answer after v2 is published.

Depending on the architecture, the decision to reuse an answer may consider document versions, the authorised scope, the version of business rules and a maximum age. These are design choices to test, rather than automatic properties of the search engine.

An expiration time limits retention. It does not necessarily replace invalidation triggered by a critical correction or a permission change. Also plan for a delayed invalidation signal: which check then prevents an answer that is no longer acceptable from being served?

## Treat deletion as an end-to-end operation

Consider a separate variation: at 09:10, C-17 is deleted from the source. The check does not end when that operation succeeds. Inspect the copies used to find the document, assemble the context and reuse an answer.

MongoDB change streams expose delete events with the document's identity. The full document is no longer present in the standard event; pre-images can be configured under specified conditions. 3

The application therefore needs to know the dependencies that identify the corresponding chunks and caches. We suggest testing deletion through the final user journey: search, a rephrased question, a cached answer and opening an earlier citation. A copy retained for archiving needs explicit rules and must not be confused with a source that can still be used to answer a question.

## Check permissions separately from freshness

Revocation is another variation, to be tested separately. C-17 still exists and version v2 is correct, but the person asking the question is no longer entitled to read it.

The answer can be completely current and still be prohibited for that person. Access checks must cover the content that could enter the context and answers already stored. Asking the model to respect permissions does not replace an authorisation decision enforced by the application.

MongoDB describes using a tenant field as a search pre-filter. 4 This is useful when the architecture includes that mechanism. The system still needs to determine the authorised tenant and handle more granular permissions. A filter freely supplied by the requester does not, by itself, constitute an access policy.

## Attach a version to the citation

A working link shows that a resource can be opened in a particular context. It does not prove that the cited passage belongs to the currently applicable version or that the recipient is entitled to read it.

For an actionable citation, we recommend keeping the document identifier, its version, the passage used and the access timestamp. The W3C PROV-O provenance model distinguishes concepts including quotation, revision and invalidation. 5

The operational goal is to answer a precise question: which state of the document produced this conclusion? A new version may require a new check. An earlier citation can remain useful for understanding a past decision when its historical context stays clearly visible.

## Prepare recovery without multiplying effects

Freshness also depends on restarting after an interruption. A consumer may apply an update, then stop before recording its progress. When it resumes, it may encounter the same event again.

Our exercise uses the fictional event e-417, which requests the move of C-17 to v2. The simulation's oracle is explicit: two attempts should produce one business effect while leaving both attempts visible in the trace. That is the exercise's expected result, not a guarantee attributed to MongoDB.

The change streams specification states that idempotent processing is the application's responsibility. 6 Define an event identity, a deduplication strategy or an idempotent write, then examine the boundary between applying the update and recording progress. External effects require their own handling.

A resume token does not replace these checks. Resuming also depends on the history that is still available in the oplog. 7

## Measure what the team needs to explain

An operational review can track lag at each stage, unprocessed changes, failed invalidations and recovery cases that require intervention. Define thresholds with the people responsible for the business process rather than copying them from an example.

For every alert, specify who intervenes, which answers may still be served and how to return to a verified state. Keep enough traces to reconstruct an answer's journey without unnecessarily exposing document contents.

The final acceptance question is simple: can we show why this person received this version at this moment? A clear answer connects engineering decisions to the business need.

## Put the chain to the test

Start with a fictional document and a stable question. Change one condition at a time: correction, deletion, revocation, then processing recovery. For each scenario, write the expected result before execution and keep the observed result separately.

This creates a reproducible review from source to answer, showing the discrepancies to address and the decisions to make.

To prepare this review with DATANALYSIS, bring a diagram of your pipeline and an example of a document whose currency matters to a decision: prepare a RAG freshness review.

# Questions & answers

**Why can a RAG answer be stale when its source is current?**

Extraction, indexing or the cache may still use an earlier version. MongoDB Search and Vector Search are eventually consistent. Trace the version at each stage and determine whether the answer was generated or reused.

**Does a cache expiration time guarantee freshness?**

Expiration bounds retention, but a critical correction or a permission change may require earlier invalidation. The required behaviour depends on the use case and must be defined and tested by the application.

**Should deletion and revocation use the same test?**

They are different scenarios. Deletion removes the source document; revocation removes a permission while the document may still exist. Examine copies, citations and cached answers separately for each scenario.

**What should be stored with a citation?**

The document identifier, its version, the passage used and the access timestamp. A working link does not establish that the passage is current or that the recipient is entitled to read it.

**What does the two-attempt e-417 exercise prove?**

It defines an expected result: two visible attempts and one business effect. It does not establish that a real system already meets that expectation or attribute an exactly-once execution guarantee to MongoDB.

**Which test should start a freshness review?**

Choose a fictional document and a stable question. Write the expected result, change one condition and record the observed result. Start with a correction, then test deletion, revocation and recovery separately.

# Sources

Documentation reviewed on 2 November 2026.

1. MongoDB Search: indexing and eventual consistency
2. MongoDB Vector Search: index considerations
3. MongoDB: delete events and pre-images
4. MongoDB: multi-tenant architecture and pre-filtering
5. W3C: PROV-O provenance ontology
6. MongoDB: change streams specification and application idempotence
7. MongoDB: resuming change streams