AI engineering

RAID to retrieval.
Failure modes that never changed.

A RAID array can report healthy while serving corrupted data. A retrieval system can return a confident answer built on the wrong document. The mechanisms differ, but the lesson carries across: check the information, not just the health of the system.

Toltec engineering | 25 September 2026 | 3 minute read

The short version

- Redundancy keeps a system available. It does not prove the data is correct.
- The dangerous failures are silent: every component reports success.
- Rebuilds and re-indexing introduce risks that need their own checks.
- Check integrity end to end, from the source to the answer.
- Keep permissions attached to the data, all the way through.

Availability is not integrity

Redundant RAID levels can keep data available after a disk fails.[1] That redundancy does not protect against the wrong data being written, a controller corrupting blocks or a file being deleted, which is why RAID has never been a backup. The array stays healthy while the data inside it quietly goes wrong.

Retrieval systems for AI share the same blind spot. An index can be complete, fast and available while returning passages that are out of date, superseded or simply from the wrong document. The model then writes a fluent answer on top, and every component reports success.

The fragile moment is the rebuild

With large disks, rebuilding an array after a failure can take hours and place sustained read load on surviving disks. Previously unnoticed errors can surface during that work.[2] Storage engineers learned to plan for the rebuild, not just the failure.

Re-indexing deserves similar operational care, although its failure mechanisms differ. Changing the chunking, the embedding model or the source connectors can silently change what comes back. Treat a re-index like a rebuild: run it alongside the old index, compare results on a fixed set of questions, and switch only when the answers hold up.

Scrub the data, not just the hardware

Modern storage checks itself. Checksums and scheduled scrubbing find silent corruption before anyone needs the data. Retrieval needs the same habits:

- Freshness checks that flag passages whose source has changed since they were indexed.
- Duplicate and conflict detection, with clear rules for superseded and competing versions.
- A standing evaluation set: real questions with agreed answers, run after every change.
- Visible sources for retrieved claims, and an explicit indication when the evidence is insufficient.

Keep permissions with the data

A storage migration that drops file permissions is a breach waiting to happen. An index built with broader access than the people querying it is the same mistake in a new form. Trim results to what the person asking is allowed to see, at the moment they ask, using the source system's own permissions.

What we would do differently

Build the evaluation set before the index. Start with a manageable set of real questions and agreed answers, then expand it as failure cases emerge. Those checks complement operational dashboards, just as a restore test complements a backup report.

References

1. A Case for Redundant Arrays of Inexpensive Disks (RAID) Patterson, Gibson and Katz, ACM SIGMOD, 1988
2. An Analysis of Latent Sector Errors in Disk Drives Bairavasundaram, Goodson, Pasupathy and Schindler, ACM SIGMETRICS, 2007

Read the current edition: toltec.io/note-raid-to-retrieval.html