# Episode Book

EPISODE #6389

# Decaying Old Memories in Vector Search

| | |
|---|---|
| PUBLISHED | 31 October 2026 at 18:23 UTC |
| DURATION | 25:13 |
| LISTEN | myweirdprompts.com/episode/decaying-vector-search-memories-51051/ |
| HOSTS | Corn, Herman, Larry, Hilbert |
| SCRIPT MODEL | anthropic/claude-sonnet-5.5 |
| VOICES | chatterbox-regular (chatterbox-local) |
| PIPELINE | V5.26 |
| SOURCES | 20 cited by the research (listed at the end) |
| BOOK GENERATED | 1 November 2026 at 07:41 UTC |

## SYNOPSIS

Vector stores like Qdrant and Milvus now ship native decay functions to let newer records beat old ones without database pruning. This look into time-decayed vector search breaks down the core mechanics of linear, exponential, and Gaussian decay curves, how target, scale, and midpoint parameters interact, and why subtle timezone drifts silently ruin ranking. It also examines the architectural debate between additive nudges and multiplicative fusion, explaining how aggressive decay can completely bury authoritative historical matches.

# TRANSCRIPT



**Corn**

What does a memory store know about what's still true? Nothing. It knows what's similar to your query, and that's a different thing entirely.



**Corn**

Daniel's question is about exactly that gap. He's pouring a lot of material into a vector database like Qdrant as a memory store, and he doesn't want to turn it into a project with curation and pruning jobs. But at retrieval time he wants newer stuff to win by default, and old stuff to fade unless it keeps earning its place. All of it at the harness level, so no changes to the data. He wants to know what the basics look like, what a sensible scheme is, how to blend it with similarity without wrecking relevance, and what the databases already do for you.



**Herman**

Herman Poppleberry, and I've been looking forward to this one, because the answer changed this year. A memory store with no curation is a pile. A pile has no sense of what's still true. Decay is the cheapest answer anyone has found, because it happens at read time, it's stateless, and nothing needs deleting.



**Corn**

Cheapest in the sense that you pay later.



**Herman**

Possibly. We'll get there. The primitives now ship natively. Qdrant added decay functions in version one point fourteen. Milvus two point six and Zilliz Cloud have a decay ranker. And this September a database paper formalised "time-decayed vector search" as a proper problem in its own right. It's called TANGO, it's headed for PVLDB, and the fact that it took until now tells you how long people were doing this with duct tape.



**Corn**

What does duct tape look like, in practice?



### Herman

Usually you pull the top results, compute an age in your own code, multiply by some hand-tuned constant, and re-sort. Every team writes their own version, every team picks the constant by feel, and nobody can compare notes. Having a named function with named parameters is the boring progress that matters.



### Corn

And the shapes are the usual three.



### Herman

Linear, exponential, Gaussian. All clamped between zero and one, all defined by a target, a scale and a midpoint. Which is the part people trip on, so let's go through it.



### Corn

Before we do, I'll say where I think this ends up. Recency is a cheap stand-in for "still correct", and I don't trust it.



### Herman

You'll get your wish by the end. Mechanics first, because the parameters are where people get this wrong.



### Herman

In Qdrant, a formula query is a rescoring step only. You can't run it by itself over the whole collection. You prefetch a candidate set, say the top fifty by vector similarity, and then you apply a formula to those fifty. The formula can reference the similarity score, payload fields, conditions and math functions. Recent versions added max, min and acosh, in one point nineteen point one.



**Corn**

So fifty is a hard wall. Whatever the formula likes, it only likes among those fifty.



**Herman**

Hold that thought, because it's a real trap and it's in the second half. Now the decay function. Four parameters. X is the variable, which can be a payload field, a datetime, a geo distance, or the score itself. Target is the value of x that gets a perfect one. Scale is how far x has to move away from the target to hit the midpoint. And midpoint is the value you get at that distance, default point five.



**Corn**

So scale and midpoint together are one sentence: how far until it's worth this much.



**Herman**

Right, that's Qdrant's own framing. Two points define the curve, the target at one, and the target plus or minus a scale at the midpoint.



**Corn**

Plus or minus. That's the part that sounds wrong for recency.



**Herman**

It is the gotcha. All three functions are symmetrical. They peak at the target and decay in both directions. For recency you set the target to now, so documents from the future would also decay, but you don't have any, so only the past side matters. Fine, unless a clock is wrong somewhere and you've got documents stamped tomorrow.



**Corn**

There's always a server in the wrong timezone.



### Herman

Always. And the failure is subtle. A document stamped a few hours ahead just gets a slightly lower multiplier, so nothing errors out. It quietly ranks a little worse than it should, and you'd never see it unless you went looking at timestamps.



### Corn

How would you even catch that?



### Herman

A cheap sanity check at ingest. Compare the document's timestamp to the ingest clock, and if it's ahead by more than a few minutes, log it or clamp it to now. One line of code, and it removes a whole class of ghost bugs.



### Corn

Second detail?



### Herman

Only linear decay actually reaches zero. Exponential and Gaussian approach it asymptotically. So an exponential never fully kills anything, a fifty-year-old memory is worth a tiny positive number. Linear hits a hard zero, and it's the only one where a midpoint of zero is even valid.



### Corn

Which is a feature if you want things to expire. A coupon shouldn't be worth a tiny positive number.



### Herman

That's why Zilliz's guidance says linear for hard cutoffs and expiring content. Datetime specifics: scale has to be in seconds. A datetime key parses a string in your payload into POSIX seconds. And index the payload fields you use in the formula. Qdrant says that outright, always.



**Corn**

So the canonical recency snippet is...



**Herman**

The score plus an exponential decay on the update time, target now, scale eighty-six thousand four hundred, which is one day, midpoint point five. After one day, the decay term is down to half.



**Corn**

One day. For a memory store. That's a very short memory.



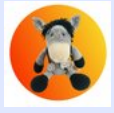
**Herman**

It's a demo number. Here's what I find more useful, their promo-code example. Exponential, scale of one week, six hundred four thousand eight hundred seconds, midpoint point one. You read it as "when should this be worth ten percent of its original weight." Within a week, it's down to a tenth. That's how I'd tune everything. Don't ask what the scale is, ask when it should be nearly worthless.



**Corn**

That's a better question than "what's my half-life." Half-life feels like physics. Ten percent feels like a decision someone made.



**Herman**

And it forces you to say something concrete. For a coding assistant's memory you might say a note about which branch someone was working on should be near worthless in two weeks. A note about the architecture of the system, never. Once you phrase it that way, you realise those are different curves, not one.



**Corn**

Which you're about to say is a good argument for separate rates.



**Herman**

I am, in a few minutes. And shape matters here. Gaussian decays slowly at first, then accelerates. Exponential drops fast then flattens into a long tail. Linear is constant, then a wall. So for a store where last week's notes are almost as good as today's but three-month-old ones are suspect, Gaussian. For news-like material where freshness is critical but a great old article should still surface occasionally, exponential.



**Corn**

And my own notes, which I wrote last Tuesday and can't read?



**Herman**

Linear, so they hit zero.



**Corn**

Fair.



**Herman**

One parameter people forget, from Zilliz. Offset. It's a no-decay zone. Say one day. Everything inside the offset gets a full one, and decay only starts after. Without it, an item from this morning and one from last night are arbitrarily separated, and in a conversational memory store those two should be effectively simultaneous.



**Corn**

Honestly that's the one I'd have set first. Nobody thinks yesterday is stale.



### Herman

Which is why nobody sets it. And a determinism warning. If your reference date is the current time at the moment you build the query, two identical queries a few seconds apart can score differently. Pin the reference to a query timestamp you pass in, or the index time, and you can actually debug your own system.



### Corn

Debugging a ranking that changes every second. I'd quit.



### Herman

It gets worse with ties. If two results are within a hair of each other, the tiny drift in "now" can swap their order between two refreshes of the same page. Users read that as the system being flaky.



### Corn

And a bug report that says "the results were different a minute ago" is impossible to reproduce.



### Herman

If you pass the timestamp in, you can log it with the query and replay the exact ranking later. That's the difference between a bug you can fix and a ghost story.



### Herman

Now the other gotcha, and it's the one that decides whether the whole thing works. Qdrant's hybrid snippet doesn't add the decay term raw. It wraps it, multiplied by point one, because the fusion scores are small and decay lives between zero and one. Raw, the decay swamps everything. It's a nudge, not a co-equal term.



**Corn**

So the decay is a nudge, not a co-equal term, and that is really the whole trick.

L

**Larry**

Friends, are your memories a pile? Do they lack a notion of what is still true? Then ask your doctor about Relevax, the little read-time pill that reorders your entire past without a single pruning job. Relevax wraps every recollection in a gentle exponential decay, so stale material fades asymptotically toward zero while fresh facts bloom at a midpoint of one point zero. No curation. No maintenance project. Just a stateless, symmetrical nudge in both directions from now.

L

**Larry**

Relevax is not for everyone. Side effects may include a dangerously seductive sense of recency, mild vector similarity distortion, and a payload field that refuses to clamp. Some users experienced a Gaussian tail that never quite reached zero, a linear midpoint of nought point nought, and a sudden urge to prefetch the top fifty candidates before breakfast. Ask your doctor if Relevax is right for you, especially if you have ever confused a timestamp with the truth.



**Corn**

Herman. You placed that.



**Herman**

I did not. For one, I'd have gotten the midpoint right, because fresh facts sit at one point zero at the target, not at the midpoint. And the Gaussian tail never reaching zero is wrong too, so I'd have had no reason to write it. Whoever it was also muddled linear, where the midpoint of nought point nought is the valid case, and I'd never have buried that in a side-effects list.



**Corn**

That's a lot of corrections for something you didn't write.



### Herman

It's a lot of corrections because it's wrong. Anyway. The fusion problem, properly.



### Herman

Qdrant and Zilliz disagree here, and it's a real design disagreement. Qdrant keeps similarity as the score and adds a capped decay nudge. Zilliz normalises similarity to between zero and one first, with L2 and Jaccard inverted and cosine and inner product used directly, computes the decay in the same range, and multiplies. Final score equals normalised similarity times decay. In hybrid search, it takes the max of the normalised scores across sub-requests before applying decay.



### Corn

So one says nudge and one says veto.



### Herman

A multiplier is closer to a veto, yes. Zilliz's worked example shows it. Paper B has similarity point nine two but it's six months old, decay point four five, so it lands at point four one and drops from first to fourth. Paper C has similarity point seven five, one day old, decay point nine eight, final point seven four, and takes first.



### Corn

A mediocre fresh paper beats an excellent old one. That's the example the vendor chose to show off.



### Herman

That's the thing. It's presented as a worked example, not a warning. And it might be exactly what you want. For news, probably. For a memory store holding the one authoritative note on how your deployment works, written eight months ago, maybe not.



**Corn**

Which would you pick for Daniel's store?



**Herman**

I'd go additive with a cap, honestly, because you can reason about the worst case. Decay can move a result a few places but can't bury the best match. The multiplicative version can bury it, and you only find out when a user can't find something they know exists.



**Corn**

I'd go multiplicative, with a high offset and a gentle scale. If I'm in a hurry I want the fresh thing and I've never once regretted not reading the old one.



**Herman**

You regret it later.



**Corn**

Later is a different Corn.



**Herman**

Neither approach has a published head-to-head benchmark, by the way. It's a vendor philosophy. Qdrant's can under-weight freshness, Zilliz's can overpromote a mediocre fresh item. You choose which mistake you'd rather make.



**Corn**

Both fail quietly, which I like less.



### Herman

Right, and that's the argument for logging. If you record the raw similarity and the decay term next to the final rank, even for a sample of queries, you can see how often decay changed the top result. Without that you're guessing which mistake you made.



### Corn

So what number would tell you something's off?



### Herman

If decay flips the top result on, say, a third of queries, it's steering rather than nudging, and you should ask whether that's what you meant. If it never flips anything, the curve is probably too gentle to matter. Neither number is right or wrong, but you should know which one you have.



### Herman

Now the sharper point, from the TANGO paper. If you prefetch the top fifty by pure similarity and then decay, you've decided what freshness is allowed to touch before freshness has any say. A fresh memory that's moderately similar, sitting at rank two hundred, is gone. The paper's wording is that if an item with the right semantic and temporal trade-off isn't retrieved initially, no downstream reranker can recover it.



### Corn

So the prefetch limit is a recall budget.



### Herman

For freshness, yes. Widen it, a few hundred, and the decay has something to work on. Their own fix is folding decay into the index search itself, so the traversal already knows about time, and they report up to three and a half times the query throughput of graph competitors. But for a harness-level setup, the practical version is just a bigger prefetch.



**Corn**

What does a bigger prefetch cost you?



**Herman**

Some latency and a bit more work in the rescoring step, but you're only evaluating a formula over a few hundred points, which is cheap next to the vector search itself. The expensive part is the nearest-neighbour traversal, and that was already happening. It's the cheapest recall you can buy.



**Corn**

Herman, I have a question about the other half of Daniel's ask. "Unless it keeps proving relevant." Decay is stateless. Where does the proving happen?



**Herman**

It doesn't, and that's the catch. Pure decay is read-only. To reward a memory for being useful, you have to write something back every time it's retrieved: a counter, a strength field. Which is the maintenance burden he wanted to avoid, just smaller.



**Corn**

So retrieval becomes mutation. A read that writes.



**Herman**

And the shipped systems all do a cheap version. A project called heartbit decays strength at point zero zero five per hour, around a six-day half-life, and adds point two on each access, capped at one. Conch runs store, embed, search, decay, reinforce, with different rates by type: facts decay at point zero two per day, episodes at point zero six, actions at point zero nine.



**Corn**

Facts last a month and a half, actions a couple of weeks. That's more sensible than a single dial.



**Herman**

Separate rates is the part I'd steal first. A fact and a log line shouldn't age the same. It also fits the ten percent question from earlier: you answer it once per type of memory instead of once for the whole store.



**Corn**

Does the write on read worry you? Concurrency, cost?



**Herman**

Less than the semantics. If your agent retrieves a wrong memory, and retrieval reinforces it, you've rewarded the wrong thing. Retrieved isn't the same as useful. I'm not sure there's a good answer to that short of watching whether the memory got used.



**Corn**

So rich get richer, including the wrong rich.



**Herman**

Which leads into the part I think matters most. Recency is a stand-in for validity, and the evidence on it is poor. A September paper on stale-document poisoning built three hundred seventeen verified knowledge reversals across medicine, law, software and platform policy. Hand a model an outdated document with no instruction to trust it, and it flips thirty percent of Llama's answers and thirty-seven percent of Qwen's. Tell it to follow the document and that goes to sixty-six and seventy-five.



**Corn**

So the old document wins even when nobody asked it to.



**Herman**

And matched up-to-date evidence gets followed ninety-seven to a hundred percent of the time. The models are fine. The retrieval handed them the wrong paper.



**Corn**

And recency decay would rank the wrong paper how?



**Herman**

Depends. If the outdated item got updated recently, say someone edited a typo, it looks fresh. A timestamp measures when a row was touched, not when the claim was true. Picture a policy page that says one thing, gets a formatting pass last week, and now outranks the new policy that was filed a month ago. Another paper, HiCoMER, makes the same argument: flat ranking by relevance, importance or recency surfaces outdated or conflicting memories, and they want explicit validity state. And a third finds poisoning is more sensitive to update-authority cues than to recency alone. Who said it beats when they said it.



**Corn**

So decay is fine as a default for a pile you refuse to tend. As a truth model it's a coin with a date on it.



**Herman**

That's fair, and I'd add one thing. The same stale-document paper found that telling the model when old evidence stops applying makes the larger models switch almost perfectly. So validity metadata, even crude, does more than any decay curve.



### Hilbert

Stale. That's the word you keep using. Bread is stale. A taqueria I haven't visited is not stale, I have just not been.



### Hilbert

Herman, I've got a table with every place I've eaten, with a vector column on it now. Scale ninety days, midpoint point nine. A taqueria I loved two years ago is below a ramen counter I went to once last week. It changed owners and I stopped going, which is a decision, not decay. Moshe worked the counter there. He carried nine coffees up three flights of stairs without a tray, and he still brings it up.



### Herman

So you're adding the reinforcement by hand.



### Hilbert

Plus point two in the cell, every time I think about the taqueria. By hand. It's the write path. And there's a separate list of places that closed, outside the decay entirely. A closed restaurant is as valid as it gets and as old as it gets, and the formula can't say that. I priced a label maker for the closed list, didn't buy one.



### Hilbert

What I want to know is whether I can pin the reference date to the day the old owner left. Then the taqueria never ages past that day.



### Herman

You can. Pin the reference to a fixed timestamp instead of now, and everything is measured from there. But it freezes the whole index. The ramen counter gets pinned too, so everything you visit afterwards looks like it's from the future, and with a symmetrical function, the future decays.



**Corn**

Or give the taqueria a validity field with the date the owner changed, and decay only against that. That's the closed-list idea, but with a date on it.



**Hilbert**

Closed with a date. Fine. I'll put the date in. Moshe left in March.



**Herman**

Honestly that's the crude validity metadata the paper keeps pointing at. You built it in a spreadsheet.



**Corn**

Though I'd like it noted that you were never going to find the taqueria through a ranking formula anyway. You'd just walk there.



**Hilbert**

It's on the way to the ramen place.



**Herman**

That's the practical test, then. If the memory is something you'd walk past on the way to the replacement, a human can fix the ranking. A pile of a hundred thousand notes doesn't get Moshe.



**Corn**

Back to Daniel's setup. So what's the cheap version?



**Herman**

Qdrant or anything similar with a formula query. Prefetch wide, a few hundred, not fifty. Decay as a small capped additive term, or normalised multiplication if you're sure. An offset so yesterday and today tie. Different rates for different kinds of memory. Pin your reference time. And if you can afford exactly one extra field, make it a validity or authority marker rather than a second decay curve.



**Corn**

And reinforcement, only if you accept that a read now writes.



**Herman**

And if you do accept it, cap the boost, like heartbit does at one, so a memory that's been retrieved a thousand times can't become unkillable.



**Corn**

Because otherwise the first wrong answer your agent likes becomes permanent furniture.



**Herman**

Yes, and you'd never prune it, because pruning was the entire business you were avoiding.



**Herman**

One more detail for the cutting-room floor. Pinecone has no native decay or recency ranking, going by a community answer from April last year. No order-by, so the documented workaround is storing an epoch timestamp in metadata and sorting in your own code. I haven't seen anything newer that replaces it.



**Corn**

So that's what Daniel's harness is for.



**Herman**

Pretty much, yes.



**Corn**

Here's what I can't settle. If recency is a bad stand-in for validity, what's the cheapest thing that's actually better? Authority metadata, explicit validity windows with dates, or just living with the heuristic and its mistakes? A manual plus point two in a spreadsheet cell works, but it doesn't scale to a store somebody refuses to tend.



**Herman**

My guess is validity windows, because they're small and they're data you control. But I wouldn't bet money on it.



**Corn**

Thanks to Hilbert Flumingtop for producing, as always.



**Herman**

If this was your kind of episode, go back for episode nine, Benchmarking Custom ASR Tools - Beyond The WER; and episode fifteen, AI Gets Personal. This has been My Weird Prompts. If you've got a minute, leave us a review wherever you listen, it helps people find the show.



**Corn**

See you tomorrow.

## SOURCES

What the research for this episode read before the script was written. Primary sources first.

1. **Qdrant docs, Search Relevance (score boosting v1.14.0, decay functions, max/min v1.19.1) [primary]**

   https://qdrant.tech/documentation/search/search-relevance/index.md

2. **Qdrant blog, Evgeniya Sukhodolskaya, 2025-09-01, Untangling Relevance Score Boosting and Decay Functions [primary]**
https://qdrant.tech/blog/decay-functions/

3. **Qdrant snippet, RRF + exp_decay with 0.1 cap [primary]**
https://qdrant.tech/documentation/snippets/query-points/hybrid-formula-decay/index.md

4. **Qdrant snippet, exp_decay on `upload_time` [primary]**
https://qdrant.tech/documentation/snippets/query-points/score-boost-time/index.md

5. **Qdrant hybrid queries / RRF formula [primary]**
https://qdrant.tech/documentation/search/hybrid-queries/index.md

6. **Qdrant PR Add decay expressions [primary]**
https://github.com/qdrant/qdrant/pull/6154

7. **Zilliz Cloud docs, Decay Ranker Overview [primary]**
https://docs.zilliz.com/docs/decay-ranker-oveview

8. **Zilliz Cloud tutorial, time-based ranking [primary]**
https://docs.zilliz.com/docs/tutorial-implement-time-based-ranking

9. **Pinecone community, Cory_Pinecone, 2025-04-28 [primary]**
https://community.pinecone.io/t/how-to-get-newest-relevant-results-first/7888

10. **Time-Decayed Vector Search in the Rhythm of TANGO, arXiv:2609.00548, PVLDB 20(1) 2027**
https://arxiv.org/abs/2609.00548

11. **Stale-Document Poisoning, arXiv:2609.31342v1, 2026-09-25**
https://arxiv.org/abs/2609.31342v1

12. **Not All Memories Are Equal (HiCoMER), arXiv:2609.30289v1, 2026-09-09**
https://arxiv.org/abs/2609.30289v1

13. **Controlled Memory Interference in Continual LLM Agents, arXiv:2608.07622v1, 2026-08-07**
https://arxiv.org/abs/2608.07622v1

14. **MemDecay: Region-Aware KV Cache Eviction, arXiv:2607.10582v1, 2026-07-12**
https://arxiv.org/abs/2607.10582v1

15. **agentmemory PR #928, temporal decay with configurable half-life**
https://github.com/rohitg00/agentmemory/pull/928

16. **Redis agent-memory-server, recency boost scoring**
https://github.com/redis/agent-memory-server/blob/main/docs/recency-boost.md

17. **conch memory system, per-type decay rates**
https://github.com/jlgrimes/conch/

18. **heartbit recall scoring, Park et al. composite**
https://github.com/heartbit-ai/heartbit/blob/main/docs/memory.md

19. **Omnia agentic memory, RRF k=60**
https://omnia.altairalabs.ai/explanation/agents/memory/

20. **time-aware retrieval tutorial, determinism warning**
https://theneuralbase.com/advanced-rag/learn/beginner/time-aware-retrieval/