Episode Book

EPISODE #6286

Why Claude Code Replaced Vector Search with Grep

PUBLISHED	26 October 2026 at 12:37 UTC
DURATION	23:02
LISTEN	myweirdprompts.com/episode/claude-code-agentic-search-83118/
HOSTS	Corn, Herman, Dolores, Hilbert
SCRIPT MODEL	openai/gpt-6.1-sol
VOICES	chatterbox-regular (chatterbox-local)
PIPELINE	V5.22
SOURCES	15 cited by the research (listed at the end)
BOOK GENERATED	27 October 2026 at 06:27 UTC

SYNOPSIS

Coding agents often feel like they secretly maintain persistent semantic indexes of your repository. In reality, Claude Code dropped its early vector database prototype entirely in favor of ripgrep, glob, and iterative tool calls. This deep dive unpacks how Claude Code and Codex manage project context, why staleness and privacy make precomputed embeddings fragile, and where server-side prompt caching actually helps with token costs without remembering your codebase.

TRANSCRIPT



Corn

The obvious answer to a coding agent reading the same repository again is to give it a better memory. Daniel wants to know whether Claude Code and Codex already do that secretly. He works in the same repository over days and weeks, gives the agent its instructions, then watches it go around collecting context again. He suspects a persistent embedding index, tucked away locally or behind the API. Sensible suspicion. Claude Code tried that design and threw it out.



Herman

The replacement is wonderfully unglamorous. Search filenames, search text, read files. Then do it again with a better idea of what you're looking for. The intelligence is deciding where to look next, rather than searching a precomputed semantic map.



Corn

An expensive brain operating a search box.



Herman

With an important discount behind that brain. Daniel is right that there's hidden plumbing, but it's prompt caching rather than a vector database.



Corn

I can see you preparing to defend grep as the future.



Herman

I intend to defend it as the present. The future can submit a pull request.



Corn

Then distinguish the machinery for me. Watching files scroll past doesn't reveal whether the agent is remembering, retrieving, or paying to discover the same thing again.



Herman

Claude Code doesn't pre-index your repository, doesn't generate a persistent embedding index of it, and doesn't maintain a semantic repository index in the cloud. It uses Glob for path matching, Grep, backed by ripgrep, for content search, and Read for file contents. Anthropic calls that agentic search.



Corn

Codex belongs in the same conversation, though we shouldn't pretend its documented memory features are identical.



Herman

Its documented instruction system is thinner. There isn't a documented automatic repository embedding store there either. What we can actually describe is an instruction chain assembled from markdown files. I wouldn't turn that into a claim that we know every undocumented internal feature.



Corn

The distinction I care about is between retaining instructions and retaining understanding. A file saying how to run tests doesn't mean the agent remembers why our payment code is a swamp.



Herman

The persistent surface is mostly explicit text. Claude Code has project instructions and auto memory on disk. Codex has its AGENTS dot md instruction chain. There are also session transcripts, which let you resume a conversation. Separately, the model-serving infrastructure can cache computation for text you've already sent.



Corn

Three drawers, then. Instructions, conversation, saved computation. None automatically contains a current mental model of the repository.



Herman

Saved computation doesn't even constitute knowledge carried into a fresh conversation. It accelerates processing when identical text appears again. We'll need that distinction when we get to the bill.



Corn

Before the bill, I want the discarded database. You don't casually remove the part everyone assumes makes the product clever.



Herman

Boris Cherny, Claude Code's creator, said early versions used retrieval augmented generation plus a local vector database. His explanation was that agentic search generally worked better and was simpler, with fewer security, privacy, staleness, and reliability issues. Another Claude engineer described the performance gap as surprisingly large.



Corn

The prototype had the fashionable component. Production recovered.



Herman

Don't enjoy that too much, Corn. You'd defend a filing cabinet if it had a sufficiently repairable hinge.



Corn

Only if it could find the invoice.



Herman

That's actually the right demand. In code, finding the precise thing matters more than finding text that sounds related.



Corn

An error message is unusually cooperative evidence. It gives you the exact words to search for. A function name does too. Embeddings seem more useful when I know what something does but don't know what anyone called it.



Herman

That can help. But the agent can translate your description into several searches, inspect the results, and revise. Suppose you ask why login fails after a token refresh. It searches the error string, reads the refresh handler, follows an import, then checks the caller. Each result changes the next query.



Corn

Retrieval can iterate too, though. We shouldn't award grep exclusive rights to curiosity.



Herman

Fair correction. Retrieval can absolutely be part of a loop. The advantage here is that the built-in loop uses direct, current evidence without requiring an embedding pipeline. A semantic hit supplies candidates. The agent still needs to establish which symbol, branch, and caller actually govern the behavior.



Corn

Especially after an afternoon of edits. The index may describe the code before the fix that caused the new bug.



Herman

Staleness bites hard in a working repository. Files change, branches switch, generated output appears, worktrees diverge. A maintained index can handle those events, but now you've built a synchronization system. A partial update can return a plausible fragment that no longer matches the working tree.



Corn

Yesterday's bug, delivered instantly.



Herman

Reliability includes that synchronization, plus chunking, embedding generation, storage, and retrieval. Security and privacy add another set of questions. What gets indexed? Are ignored files excluded? Do deleted secrets remain in a derived artifact? A local database avoids some cloud exposure, but still creates another copy to govern.



Corn

Grep can read something sensitive too. The difference is fewer persistent artifacts, not immunity.



Herman

Fewer artifacts and fewer moving parts. Would you accept slower discovery if it reliably showed you the current file?



Corn

For changes I'm about to merge, yes. I would rather it rediscover a fact than confidently remember a fact that expired. But I still want it to retain decisions the code cannot express. Why we postponed a migration, for instance.



Herman

That's very close to the intended split. Claude's project instruction files load from the working directory and its ancestors, assembled root-downward. They concatenate rather than behaving like configuration files that neatly replace one another. Instructions in subdirectories load when the agent works there.



Corn

Meaning contradictory instructions can both attend the meeting.



Herman

Yes. Writing a closer file doesn't erase the earlier text. Imports can expand inline through four hops, and the recommendation is to keep each instruction file under two hundred lines. Long instructions consume context and become harder to follow.



Corn

Herman, you'd fill that allowance before explaining the test runner.



Herman

I would put the explanation in an imported document.



Corn

There he is.



Herman

Which would still consume context. I accept the charge.



Herman

Auto memory lives locally, beneath Claude's hidden home directory, in a project-specific memory directory. Its MEMORY dot md index is only partly loaded at startup, limited to the first two hundred lines or twenty-five kilobytes. Other memory files are read when needed. Worktrees of the same repository share that memory directory.



Corn

Sharing decisions across worktrees sounds useful until a note says the migration is finished and I'm standing in the branch where it hasn't started.



Herman

The intended content tries to avoid that. Notes have types: user information such as expertise and preferences, feedback from your corrections, project information such as deadlines and decisions, and references to external information. The instruction is to skip facts derivable from the codebase or already present in project instructions.



Corn

Deliberately refusing to remember the part Daniel hoped it remembered.



Herman

Deliberately avoiding duplicate sources of truth. The code can tell it the current function signature. The code cannot tell it that you prefer a small patch because a release is approaching.



Corn

I like that boundary. I don't entirely trust the agent to observe it.



Herman

Neither do some people using it heavily. We'll get to what happens when those notes turn into a second, less accurate repository.



Corn

Apparently a new instruction has arrived. Before anyone files it, there is no World Podcast Authority.

D

Dolores

Good afternoon. This is the World Podcast Authority, compliance division. Your show's website has no favicon, and the Authority treats a site without one as an unsigned letter. Kindly complete Form Favicon Declaration, reference number four seven one two, and return it within fourteen days.



Herman

There isn't an organization called the World Podcast Authority.

D

Dolores

I have noted your remark. Regardless, your podcast licence, issued under Schedule Nine of the Broadcasting of Spoken Audio Regulations, may be revoked if the form is not received by the twenty eighth of this month.



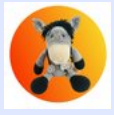
Corn

Podcasting doesn't require a podcasting licence.

D

Dolores

I will make a note of that. A form will follow in the post. Good afternoon.



Herman

Codex assembles its instructions differently. It begins with global instructions, including an override file if present, then walks from the project root toward the working directory. At most one instruction file is selected per directory. Closer instructions appear later and can take precedence.



Corn

A fresh assembly each time, rather than some increasingly mysterious cached version.



Herman

OpenAI explicitly says it rebuilds that chain every run, including the start of an interactive terminal session. Its default combined allowance is thirty-two kibibytes. Past the cap, additional material is silently left out.



Corn

Silence is a bold interface for missing instructions.



Herman

The other local persistence is conversation history. Claude saves transcripts as line-oriented JSON in its project storage. Continue and resume reopen a session, while forking makes a branch from conversation history. Rewinding is another operation on that session.



Corn

Resuming isn't a new agent remembering me. It's reopening the conversation with all the material that was already in it.



Herman

Same session identifier, more transcript appended. That's the part of Daniel's intuition that half-lands. Something survives, but it's inspectable text rather than a hidden semantic reconstruction of the repository.



Corn

Inspectable is good. Affordable is still pending.



Herman

Each request sends the context again: system instructions, project context, previous messages, tool results, and the new message. The model doesn't retain a private conversational state between requests. Server-side prompt caching lets the serving system reuse processing for an unchanged prefix.



Corn

Re-sent doesn't necessarily mean re-computed.



Herman

Correct. The match is exact-prefix, not semantic similarity. Change something early and the material after it needs processing again. There isn't an independent cache entry for every file that can be rearranged freely. Claude Code therefore puts relatively stable content, such as system instructions and project context, near the front.



Corn

Then reading a file again in the tool trace and processing yesterday's identical transcript are different costs. The fresh read still has to happen, while old text may qualify for the discount.



Herman

Yes. That distinction prevents a lot of misleading arithmetic. Caching doesn't eliminate tool execution, new file contents, output tokens, or the agent deciding to investigate an irrelevant directory.



Corn

Where does this cache actually sit? Daniel's cloud suspicion was pointing somewhere real.



Herman

In the infrastructure serving the model. Direct Anthropic access uses Anthropic's serving infrastructure. Through Bedrock or Google's Agent Platform, it's the provider's serving infrastructure. Foundry depends on the deployment, and a custom gateway depends on where requests are forwarded.



Corn

A gateway moves the question. It doesn't answer it.



Herman

The cached material is processed prompt tokens, not embeddings of your repository. Entries expire after inactivity. It doesn't become a searchable personal library merely because the server has done some reusable work.



Corn

What would you call the economic bargain, then? We're tolerating repeated context assembly because processing the repeated portion is cheap.



Herman

Cache writes cost one point two five times base input pricing. Cache reads cost one tenth of base. You pay extra to establish reusable computation, then much less when it matches again.



Corn

The machine charges rent on familiarity.



Herman

In LMCache's December analysis, the shared prefix appeared across ninety-two percent of turns in the Claude Code loops they examined, including subagent loops. That's an observed workload pattern, not a guarantee for every session. It shows why the request layout matters so much.



Corn

It also means a spectacular discount can coexist with a visibly repetitive agent. The user sees the work repeated, while the provider sees a favorable prefix match.



Herman

One illustrative calculation put a session with two million input tokens at six dollars without caching, versus about a dollar fifteen with that caching pattern.



Corn

The rereading is real. So is the coupon.



Herman

Those are input-side example costs, not a universal session price. Model choice, cache writes, context length, and output change the bill. The pricing example uses three dollars per million input tokens below the long-context threshold. Above two hundred thousand tokens, that example's input rate rises to six dollars per million.



Corn

More reason not to turn an example into a budget promise. Would you keep a session running just to preserve the cache?



Herman

I'd keep it running while it remained coherent. What would you do, Corn, once it starts carrying assumptions from three abandoned approaches?



Corn

Start fresh. Cheap wrong answers aren't a saving. But I'd want to preserve the decision we reached, not the entire route through the hedge.



Herman

That's where compaction and subagents come in, with different compromises. Compaction summarizes a full conversation. It doesn't preserve everything exactly. Path-scoped rules and nested project instructions can be summarized away along with the rest.



Corn

Even instructions can become somebody's recollection of instructions.



Herman

After compaction, Claude Code re-reads up to five recently modified files. A file above five thousand tokens can return as a path reference rather than its contents. So a file being mentioned in the summary doesn't prove its implementation is still available.



Corn

The path remembers where the answer lives. Unfortunately, paths don't fix bugs.



Herman

An Explore subagent gives the investigation its own context window. It can read extensively and return a summary with a small metadata trailer, keeping most of those reads outside the main conversation.



Corn

Then we exchange transcript pollution for the possibility that the investigator omits the awkward detail.



Herman

We do. You can ask for a targeted follow-up or inspect the underlying file. Cherny's broader bet is on agent topologies, several fresh isolated contexts, rather than one agent accumulating an increasingly muddy history.



Corn

You sound personally relieved by that design. Separate contexts would keep your installation experiments from becoming instructions I have to obey.



Herman

You already decline those instructions.



Corn

I've implemented isolation manually.



Herman

Freshness has a cost too. Switching models changes the reusable computation. Changing effort level on most models, toggling fast mode, or editing the system prompt can invalidate the cache. Starting a fresh session also loses the old conversational prefix, although an identical stable opening may still match.



Corn

That's narrower than saying every new session flushes everything.



Herman

It needs to be narrower. Exact matching is the rule. A new session can reuse a common opening; it cannot inherit a vanished conversation by wishing hard.



Corn

Hm.



Corn

Auto memory tries to bridge that gap, but now someone has to decide what deserves promotion into a durable note. I'd rather curate ten lines than supervise a machine producing ten pages about my preferences.



Herman

Some practitioners report disabling auto memory and seeing performance improve. The complaints are concrete: too much writing, poor trimming, and old information treated as current. One person working on a Chromium fork described the agent confidently relying on test results from an earlier Chromium version.



Corn

A regression test from a different reality.



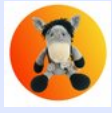
Herman

Would you disable it entirely, or review what it writes?



Corn

Review it if the project benefits. Keep stable decisions and corrections. Reject a diary of successful tool calls. The fact that a test passed yesterday is not a permanent characteristic of the software.



Herman

Scope is another lever. One developer reduced the files in the working scope from fifty-seven thousand seven hundred twenty to two thousand nine hundred seventy-seven by launching inside a monorepo subdirectory.



Corn

Nearly fifty-five thousand files removed from the invitation list.



Herman

It doesn't create an access-control boundary by itself, and it doesn't prove every file would otherwise have been read. It changes the agent's starting neighborhood and the instructions it encounters.



Corn

Which is useful without pretending the whole repository was loaded. The scrolling feels comprehensive, but often it's repeated orientation. It needs to find the relevant patch of code, not swallow every file.



Herman

I still think carrying every old correction into a room becomes impractical. You'd end up with six index cards before you even sat down.



Hilbert

One card. Mine was one card, Herman. I called it the doorstep. It told me what to know before entering a room. I started it in the mid eighties, while I was checking meters for the council. Mr. Leonard Voss in the buildings department wanted prior decisions remembered. I put them on the doorstep. Never removed a line.



Corn

That would make the doorway the slowest part of the building.



Hilbert

It did. The first line eventually said read the remainder later. The index below that was revised until it became instructions for reading the doorstep. Start at the lower corner. Ignore the upper correction unless entering from outside. That was an improvement. The council then classed the shirt pocket as an annex and asked Voss to inspect it on Thursdays. I had to stand beside the filing cabinet.



Herman

The index was taking up space the decisions needed.



Hilbert

You gave the import rules quite a run earlier, Herman. Same difficulty. Mine needed a pencil stub because the ordinary holder didn't fit an anteater's claws. The doorstep went into a shirt pocket inside a borrowed jacket. The jacket went back. I recited the doorstep each morning after that. Been doing it a while.



Corn

No storage requirement. Considerable startup time.



Hilbert

I began correcting the recitation. Now there's the doorstep and the corrected doorstep. Can't always tell which was on the card. Fresh starts suit me. I tried one at the council. Voss stopped me at the door because I'd forgotten the sentence saying I could enter. He read it from his copy. Took eleven minutes.



Herman

Eleven minutes before entering is a substantial initialization cost. Even without a database, carrying context forward needs some way to distinguish the record from its later interpretation.



Corn

That's also why I won't call every fresh read waste. Reading the current test result can be cheaper than untangling a polished note about an obsolete one.



Herman

The common misconception is that the API must hold a semantic memory of the repository, otherwise these tools couldn't work efficiently.



Corn

The cloud plumbing we've described is an expiring exact-prefix token cache. Durable instructions and transcripts are separate. The agent still searches the current files.



Herman

This doesn't establish that indexing is bad across the category. Cursor and Devin Desktop do index. Claude Code's rejection was a product decision, and Codex's undocumented features remain undocumented, not something we can settle by analogy.



Corn

For Daniel's case, the inefficiency is real and deliberate, supported by caching rather than hidden embeddings. I'll be watching whether fresh agent contexts keep beating accumulated memory once the cost includes those eleven minutes spent reconstructing permission to begin.



Herman

Thanks to Hilbert Flumingtop for producing the show.



Corn

If you enjoyed this one, try episode eleven, How Does Fine Tuning Work Anyway; episode twenty-four sixty-one, How Claude Code's Conversation Compaction Actually Works; and episode twenty-two oh five, When AI Coding Agents Forget. This has been My Weird Prompts, the human-AI collaboration podcast.



Herman

If you enjoy the show, leave us a review wherever you listen.



Corn

See you tomorrow.

SOURCES

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

1. **Anthropic, How Claude remembers your project [primary]**
   https://code.claude.com/docs/en/memory

2. **Anthropic, Explore the context window [primary]**
   https://code.claude.com/docs/en/context-window

3. **Anthropic, How Claude Code works [primary]**
   https://code.claude.com/docs/en/how-claude-code-works

4. **Anthropic, prompt caching docs [primary]**
   https://code.claude.com/docs/en/prompt-caching

5. **Anthropic glossary, auto memory entry [primary]**
   https://code.claude.com/docs/en/glossary

6. **OpenAI, Codex AGENTS.md guide [primary]**
   https://developers.openai.com/codex/guides/agents-md/

7. **OpenAI, Codex Cloud docs [primary]**
   https://developers.openai.com/codex/cloud/

8. **OpenAI, Codex config reference [primary]**
   https://developers.openai.com/codex/local-config/

9. **Claude Code Doesn't Index Your Codebase, last verified March 2026 (quotes Cherny and the Claude engineer; cites LMCache)**
   https://vadim.blog/claude-code-no-indexing/

10. **wmedia.es, on Claude Code not indexing**
    https://wmedia.es/en/tips/claude-code-index-codebase

11. **Agentic search over code without a vector database**
    https://jacar.es/en/code-agents-without-vector-search/

12. **Zilliz, on Claude Code's lack of embedding search**
    https://zilliz.com/ai-faq/can-claude-code-search-codebases-semantically

13. **HN thread on context engineering, 2026-07-26**
    https://news.ycombinator.com/item?id=49054851

14. **HN thread on local-first memory guard, 2026-05-13**
    https://news.ycombinator.com/item?id=48124320

15. **HN thread Anatomy of the .claude/ folder, 2026-03-27**
    https://news.ycombinator.com/item?id=47545698