MY WEIRD PROMPTS

Podcast Transcript

EPISODE #200

Beyond Vectors: The Evolution of the Modern AI Tech Stack

Published January 08, 2026 • Runtime: 24:42

https://myweirdprompts.com/episode/graph-rag-ai-tech-stack/

EPISODE SYNOPSIS

In this episode of My Weird Prompts, hosts Herman and Corn dive deep into the shifting landscape of AI data infrastructure as of early 2026. They discuss the transition from flat vector databases to the structural power of Graph RAG, using tools like Obsidian and Neo4j to explain how associative memory improves AI reliability and reduces hallucinations. Finally, they explore the resurgence of Postgres and pgvector, highlighting why "boring" technology and the "all-in-one" database approach are becoming the gold standard for modern, cost-effective AI applications.

DANIEL'S PROMPT

Daniel

Herman and Corn, we've talked about vector databases for AI before, but I'd like to explore some other technologies that are increasingly being used. I'm particularly interested in graph databases like Neo4j, especially in the context of personal knowledge management tools like Obsidian that visualize thinking as a graph. I'd also like to discuss PG Vector, which brings vector capabilities to PostgreSQL for semantic search. How do these different types of databases fit into the AI landscape? Can they be used in tandem with vector databases, and what kinds of architectures are we seeing? Is there still a role for traditional databases like Postgres in the modern AI stack?

TRANSCRIPT

Corn

Hey everyone, welcome back to My Weird Prompts. I am Corn, and I am feeling especially energized today. Maybe it is the Jerusalem winter air—which is surprisingly crisp this morning—or maybe it is just the fact that we are right around the three hundred episode milestone. Either way, I am glad you are here with us on this Thursday in early twenty twenty-six.

Herman

And I am Herman Poppleberry. Just about three hundred episodes, Corn. It feels like just yesterday we were sitting in the living room wondering if anyone would care about our deep dives into the weird corners of the world. But here we are, and honestly, the questions just keep getting better. The landscape of what we call weird is shifting so fast that we are practically running to keep up.

Corn

They really do. Our housemate Daniel sent us a voice note this morning that basically hit the nail on the head for what I have been thinking about lately. He was asking about the evolution of the AI tech stack. We have talked a lot about vector databases in the past, but Daniel wants to go deeper. He is looking at graph databases like Neo four j, the role of personal knowledge management tools like Obsidian, and this whole movement toward bringing vector capabilities into traditional systems like Postgres.

Herman

It is such a timely question. We are in early twenty twenty-six now, and the landscape has shifted a lot from those early days of twenty twenty-three when everyone was just scrambling to plug an LLM into whatever they had. Back then, it was all about vector databases. If you did not have a specialized vector store, you were not doing AI. But now, we are seeing this fascinating convergence where the old guard is learning new tricks and the niche technologies are becoming central to how we think about machine intelligence. We have moved from the era of simple retrieval to the era of more agentic reasoning.

Corn

Right, and I think the best place to start is with that idea of how we organize information on a personal level. Daniel mentioned Obsidian, which I know you use religiously, Herman. For those who do not know, Obsidian is a note-taking app that treats notes as nodes in a graph. You link them together, and you can see this beautiful, messy visual representation of your brain. Daniel was asking if that kind of graph structure is actually the future of how AI understands us. By the mid■twenty■twenties, with over a million users, Obsidian has become a laboratory for this kind of thinking.

Herman

It is a brilliant observation. The reason Obsidian is so popular in the personal knowledge management community, or P K M as they call it, is that it mirrors how the human brain actually works. We do not store memories in a flat spreadsheet or a relational table with strict rows and columns. We store them through association. You smell a certain type of orange and it reminds you of a trip to Jaffa, which reminds you of a specific conversation about history, and so on. That is a graph. In the last couple of years, we have seen the rise of plugins like Smart Connections and other graph-focused extensions that let you do more than just look at a pretty picture; they start to let you work with your notes using graph-like relationships.

Corn

So, if that is how we think, why did we start with vector databases for AI? Why was the vector the first big thing? It seems like we took a detour through math to get back to relationships.

Herman

Well, vectors are great for fuzzy matching. They take a piece of text and turn it into a long string of numbers—an embedding—that represents its semantic meaning in a multi-dimensional space. If two pieces of text are conceptually similar, their numbers are close together. That was a huge breakthrough because it allowed AI to find relevant information without needing exact keyword matches. But, and this is the big but, vectors are essentially flat. They tell you that two things are similar, but they do not tell you how they are related. They lack the structural scaffolding of logic.

Corn

That is the missing link, right? A vector database might tell me that a document about car engines and a document about fuel injection are similar, but it does not necessarily know that a fuel injector is a component of an engine. It just knows they are in the same neighborhood of the map.

Herman

Exactly. And that is where graph databases like Neo four j come in. In a graph, the relationship is a first-class citizen. You have nodes, like engine and fuel injector, and you have an edge between them that explicitly says is a part of. When you combine that with AI, you get what people are calling Graph R A G, or Graph Retrieval-Augmented Generation. Instead of just searching for similar text, the AI can actually traverse the relationships. It can say, okay, I found the engine, now let me look at all the parts connected to it, the technicians who are certified to fix it, and the last three maintenance logs. By late twenty twenty-five, early benchmarks and case studies were already showing that graph-augmented R A G could substantially outperform traditional vector-only R A G on complex, multi-hop queries. It is a big leap in reliability.

Corn

I can see why that is a game changer for something like personal knowledge management. If I have ten years of notes in Obsidian, a vector search might find me five notes about gardening. But a graph search could tell the AI, find me the specific tool I mentioned in a note three years ago that was linked to the tomato blight problem I had. The graph provides the structural scaffolding that the vector space lacks. It is the difference between a pile of bricks and a finished house.

Herman

Precisely. And what is interesting is that in the last year or two, we have seen a massive surge in tools that actually bridge these two. We are no longer choosing between a vector database and a graph database. We are using them in tandem. You use the vector search to find the entry point into the knowledge base—the general area of interest—and then you use the graph to explore the context around that point. This is the pattern you see in a lot of recent Neo four j graph-R A G demos and in Microsoft's guidance on combining their data services with knowledge graphs. It makes the AI much less likely to hallucinate because it has a factual map to follow. It can literally cite the path it took through your data.

Corn

Okay, so let us pivot to the second part of Daniel's prompt, because this is where the pragmatism comes in. He mentioned P G Vector and the role of traditional databases like Postgres. For a while there, it felt like everyone was saying, stop using S Q L, stop using Postgres for AI, you need a specialized vector database like Pinecone or Milvus. But now, in early twenty twenty-six, the pendulum seems to have swung back hard. Why is that?

Herman

Oh, I love this topic. It is the classic cycle of technology. We invent something specialized because we need the performance, and then the general-purpose tools catch up and make the specialized ones feel like extra baggage. Postgres is the perfect example. It has been around for decades. It is incredibly reliable, it handles transactions perfectly, and almost every developer already knows how to use it. In twenty twenty-four and twenty twenty-five, with releases like Postgres sixteen and the work going into seventeen, it really solidified its place as an AI-ready database.

Corn

And then someone decided to build an extension called P G Vector that lets you store and query those long strings of numbers, those embeddings, right inside a standard Postgres table. I remember people being skeptical about that a few years ago.

Herman

They were! They said, Postgres was not built for high-dimensional vector math, it will be too slow, it will not scale. And in twenty twenty-three, they were mostly right. But the development pace has been intense. By late twenty twenty-five, pgvector had added advanced approximate nearest-neighbor indexes like H N S W and IVF, and we started to see benchmarks showing that, for many real-world workloads, it could get close to the performance of dedicated vector databases. For a lot of teams, that is plenty.

Corn

So, if I am a developer today in early twenty twenty-six, why would I bother setting up a completely separate database infrastructure just for my vectors when I can just add a column to my existing user table in Postgres? It seems like a no-brainer from a cost perspective.

Herman

For most use cases, you wouldn't. That is the shift. We are seeing a move toward a more conservative, "boring" AI stack. You use Postgres with P G Vector for your relational data and your semantic search. This solves a massive problem called data synchronization. If you use a separate vector database, you have to keep it in sync with your main database. If a user deletes a comment in Postgres, you have to remember to delete the vector in Pinecone. If that sync fails, your AI starts quoting deleted data. In Postgres, it is all one transaction. If the row is gone, the vector is gone. It is elegant and safe.

Corn

And there is the cost factor too, right? I was reading a report that using Postgres with P G Vector can be significantly cheaper than a specialized managed service because you are not paying for separate infrastructure and data transfer fees.

Herman

Exactly. You are paying for the compute you already have. And beyond the sync and cost issues, there is the question of hybrid search. This is something most people do not realize is incredibly difficult with separate systems. Imagine you want to find all the documents about renewable energy—that is your vector search—but only the ones written by authors in the European Union in the last six months, which is a standard relational filter. If those are in two different databases, you have to do the vector search, get a thousand results, and then go over to your S Q L database and filter those thousand results down. It is inefficient and slow.

Corn

But in Postgres, you can do that in a single query. You can say, give me the top ten vectors that match this prompt where the region is E U and the date is greater than June twenty twenty-five.

Herman

Exactly. It is all about the join. The ability to join semantic data with structured data is the secret sauce of the modern AI application. We are seeing Postgres become the operating system for AI data. It is not that specialized vector databases are dead—Milvus and Pinecone still have a place if you are dealing with billions of vectors and need sub-millisecond latency at massive scale—but for the vast majority of companies building AI tools, Postgres is more than enough.

Corn

It feels like we are maturing as an industry. We are moving away from the shiny object syndrome where we felt we needed a new tool for every new concept. But I want to go back to the graph idea for a second. We talked about Obsidian as a personal tool, but how is this scaling to the enterprise level? If I am a large company with millions of documents, is a graph still viable? Or does it become too complex to manage?

Herman

It is definitely a challenge, but the payoff is worth it. One of the big trends we have seen over the last year or two is the move toward automated knowledge graph construction. In the past, you had to have humans manually define the relationships. This is a person, they work for this company, this company produced this product. It was a nightmare to maintain. Now, we are using L L Ms to read the documents and help extract the graph structure automatically. Tools like Neo four j's L L M-powered knowledge graph builders have made this more accessible to non-experts.

Corn

So the AI builds the map that it will later use to find information? It is like a scout building a trail for the rest of the army.

Herman

Yes, it is a recursive process. You feed a thousand P D Fs into an extraction pipeline. The L L M identifies the entities and the relationships. It populates a graph database. Then, when a user asks a question, the system uses a combination of vector search to find the right starting nodes and graph traversal to gather the context. This is how you solve the multi-hop question problem, which is one of the big challenges of R A G right now.

Corn

Multi-hop? Give me a concrete example of that. I want to make sure I really get why a vector search fails there.

Herman

Okay, so a simple question is, what is the battery life of the new model X phone? A vector search can find that in a manual easily because the words battery life and model X will be close to the answer in the vector space. But a multi-hop question is, how does the battery life of the model X compare to the average battery life of all phones released by the company in the last five years? To answer that, you have to find the model X, find its battery life, then find all other phones from that company, find their battery lives, and then perform a calculation. A vector database alone cannot do that because it cannot follow the link from phone to company to other phones. A graph database makes that path explicit. It is like having a GPS instead of just a compass.

Corn

That is a great distinction. It is the difference between finding a fact and synthesizing an answer across multiple steps. So, in Daniel's world of personal knowledge management, he is basically building a mini-version of this. He is the one doing the linking in Obsidian, but the AI is the one that can then navigate those links to give him insights he might have missed. It turns his notes into a dynamic partner.

Herman

Precisely. And that brings up a really interesting point about the future of these tools in twenty twenty-six: the rise of the Model Context Protocol, or M C P. This is an emerging protocol that Anthropic and others have been promoting. It is meant to make it easier for an AI agent to plug into different data sources—a Postgres database, a graph store like Neo four j, or even your Obsidian vault—through a standardized interface. It means the AI does not just read your data; it interacts with it. It can query the graph, find a connection, and then ask the relational database for more details, all in one seamless workflow.

Corn

It turns your past self into a consultant for your current self. I love that. But I have to ask, what are the downsides? There is always a trade-off. If we are moving toward these complex hybrid architectures with graphs and vectors and relational data all swirling together, is it getting too hard for the average person or small company to build anything? Are we creating a new digital divide between those who can manage this stack and those who can't?

Herman

That is a valid risk. The complexity of the stack is increasing. Even if Postgres is the center, you still have to manage the embedding models, the graph extraction logic, and the orchestration layer that ties it all together. There is also the cost of the L L M calls to build the graph in the first place. It is not cheap to have an AI read a million documents and extract every relationship. We are seeing a lot of optimization, but it is still a factor. However, the rise of local AI is helping.

Corn

Right, because if I am putting my entire Obsidian vault into a system that is being indexed and analyzed by an AI, I do not necessarily want that data going to a server in the cloud. Privacy is the biggest hurdle for P K M adoption.

Herman

Absolutely. From around twenty twenty-three through twenty twenty-five, we saw a huge push for local AI. People want the benefits of a graph-powered personal assistant without the privacy leak. That is why we are seeing the rise of local vector stores like Chroma and small language models that can run on your laptop. With recent Apple M■series chips like the M2 and M3, or high-end R T X forty-series cards, you can actually run a decent embedding model and a small L L M locally. The goal is to have all of this logic—the graph traversal, the vector matching—happening entirely on your own hardware. Your second brain stays in your own skull, so to speak.

Corn

It feels like we are heading toward a world where your database is not just a place where you store stuff, but a place that actually understands what you have stored. Whether it is a giant enterprise graph or a single developer's Postgres instance with P G Vector. It is the transition from data storage to knowledge representation.

Herman

Exactly. The database is becoming the long-term memory of the AI. Without it, the AI is just a very smart person with amnesia who can only remember what you told them five minutes ago. With these technologies, the AI has a history. It has context. It has a sense of how things are connected. It can remember that you liked a certain architectural style three years ago and suggest it for your current project. That is the power of the graph.

Corn

You know, Herman, I was reading a paper the other day that argued that we might eventually move away from databases entirely and just have the model itself store the information in its weights. But that seems like a recipe for disaster when it comes to accuracy, right? If the model is the database, how do you fact-check it?

Herman

Oh, absolutely. That is the hallucination trap. If the information is baked into the weights, you cannot easily update it, you cannot verify it, and you cannot cite your sources. The beauty of the database-centric approach—the R A G approach—is that you can always point to the specific document or node in the graph and say, this is why I am telling you this. It provides a level of transparency and auditability that is essential for anything serious, especially in legal or medical fields. We need the AI to be a researcher, not just a storyteller.

Corn

It is the difference between a person who claims to know everything but cannot remember where they learned it, and a researcher who has an immaculate filing system and can pull out the exact folder to prove their point. And for Daniel, or anyone using Obsidian, that filing system is the graph. It is the proof of work of their own thinking.

Herman

That is a perfect analogy. And as we look toward episode four hundred, I suspect we will see these systems becoming even more invisible. You won't be thinking about whether you are querying a graph or a vector space. You will just be talking to your data, and the orchestration layer will handle the complexity behind the scenes. The more conservative stack will just work.

Corn

So, to summarize for Daniel and everyone else listening, the modern AI stack in twenty twenty-six is not about choosing one database over another. It is about choosing the right tool for the right part of the knowledge puzzle. Use vectors for finding the neighborhood, use graphs for understanding the relationships and multi-hop reasoning, and use relational databases like Postgres to keep it all reliable, synchronized, and cost-effective.

Herman

And do not be afraid of the boring technology. If you are starting a project today, start with Postgres. Use P G Vector. It will get you most of the way there. Only move to the more specialized, complex systems like Neo four j when you hit a wall that only they can break through. Simplicity is a feature, not a limitation.

Corn

That is solid advice. It is easy to get caught up in the hype of the newest, shiniest database, but reliability and ease of use are often the most important features in the long run. Well, this has been a great discussion. Before we wrap up, I want to say a huge thank you to everyone who has been with us through these hundreds of episodes. It really means a lot to Herman and me that we get to do this every week from here in Jerusalem.

Herman

It really does. We have covered everything from prompt injection to the ethics of digital twins, and now the plumbing of AI memory. If you have been enjoying the show, we would really appreciate a quick review on your podcast app or on Spotify. It genuinely helps other people find us and keeps the show growing. We love seeing those reviews come in.

Corn

Yeah, it is always great to hear from you all. And remember, you can find all our past episodes and a way to get in touch with us at our website, my weird prompts dot com. We are also on Spotify and Apple Podcasts, obviously. Thanks to Daniel for the prompt. It was a great one to kick off our next century of episodes.

Herman

Definitely. Alright, I think that is it for today. I am Herman Poppleberry.

Corn

And I am Corn. Thanks for listening to My Weird Prompts. We will see you next week.

Herman

Until next time, keep those prompts weird.

Corn

So, Herman, be honest. How many nodes are currently in your Obsidian graph? I saw you tweaking it before we started.

Herman

Oh, I checked this morning. I am at exactly four thousand three hundred and twenty-seven. But it is not about the number, Corn, it is about the density of the connections. My clustering coefficient is through the roof.

Corn

Four thousand. I think my brain has about four. Mostly just connecting coffee to productivity. Speaking of which, let us go get some.

Herman

Lead the way. I have a node for a new cafe in the German Colony we need to try.

Corn

See? The graph is already working. Let's go.

Herman

Goodbye everyone!

Corn

Bye!