MY WEIRD PROMPTS

Podcast Transcript

EPISODE #263

Beyond the Table: Why AI is Moving to Graph Databases

Published January 21, 2026 • Runtime: 19:01

https://myweirdprompts.com/episode/graph-vs-vector-databases/

EPISODE SYNOPSIS

In this episode of My Weird Prompts, Herman and Corn dive deep into the digital plumbing of 2026 to answer a pressing question: is the era of the relational database finally coming to an end? Sparked by a prompt from their housemate Daniel, the brothers break down the fundamental differences between the rigid tables of SQL, the semantic "neighborhoods" of vector databases like Pinecone, and the relationship-first architecture of graph databases like Neo4j. Herman explains the technical magic of the "edge" and why index-free adjacency is the secret to scaling complex queries. They also explore the rise of GraphRAG—a powerful combination that uses knowledge graphs to ground AI models in factual truth, effectively ending the reign of LLM hallucinations. From the "join penalty" to the future of polyglot persistence, this discussion provides a comprehensive roadmap for anyone looking to understand how data is being restructured for the age of artificial intelligence. It’s an essential guide for developers navigating the shift from being "mechanics" of code to "urban planners" of information.

DANIEL'S PROMPT

Daniel

I'd like to chat about graph databases. Everything is changing so fast with the advent of AI, and we've learned that AI wants to be vector-native for semantic retrieval, which shifts attention to things like Pinecone and vector database storage. I've been looking at Neo4j, which makes a lot of sense because it uses nodes and relationships rather than linear storage. As our requirements for data storage grow exponentially, do you think the backbone of data storage will shift toward vector-native formats and graph databases, or will SQL and vector databases continue to play separate roles? Also, what are 'edges' and how is this all going to play out?

TRANSCRIPT

Corn

Hey everyone, welcome back to My Weird Prompts. I am Corn, and I am joined as always by my brother, the man who probably knows more about data structures than he does about his own family history.

Herman

That is Herman Poppleberry to you, Corn. And honestly, considering our family history is a bit of a distributed mess, a structured database might be the only way to make sense of it.

Corn

Well, before we dive into the digital plumbing of the world, we have to address the prompt that started this. Our housemate Daniel sent this one in while nursing a pretty gnarly cold. He is apparently self-medicating with a combination of oranges and white wine.

Herman

It is a bold therapeutic choice. I am not sure if the vitamin C balances out the Chardonnay, but it certainly makes for an interesting morning in the kitchen.

Corn

He sounded a bit congested, but his brain is clearly firing on all cylinders because he is asking about the backbone of modern data storage. Specifically, he is looking at the collision between graph databases like Neo-four-j and the rise of vector-native storage like Pinecone. He wants to know if S-Q-L is going to be left in the dust and, crucially, what on earth an edge is.

Herman

This is such a timely question for January twenty-first, twenty-twenty-six. We are at this fascinating inflection point where the way we store information is finally catching up to the way we actually think. For decades, we forced our data into these rigid, rectangular boxes called tables. But A-I does not think in tables. It thinks in relationships and high-dimensional space.

Corn

Right, and Daniel mentioned that transition from S-Q-L to No-S-Q-L like Mongo-D-B, and now to vector-native. But I want to start with the graph part because that seems to be where a lot of the hidden power lies. Herman, for the uninitiated, let's tackle Daniel's question directly. What is an edge?

Herman

In the world of graph databases, you have two main components: nodes and edges. Think of a node as a noun. It is a person, a place, a product, or a concept. If you were building a graph of our house, you and I would be nodes. Daniel would be a node. Jerusalem would be a node.

Corn

Okay, so the nodes are the entities. What about the edges?

Herman

The edges are the verbs. They are the relationships. An edge connects two nodes and describes how they interact. So, there is an edge between you and me that says brother of. There is an edge between us and this house that says lives in. In a traditional S-Q-L database, those relationships are often hidden in foreign keys or joining tables. You have to do a lot of math and searching to reconstruct the connection. But in a graph database like Neo-four-j, the edge is a first-class citizen. It is stored physically as a pointer to the next piece of data.

Corn

That is a key distinction, right? I remember we talked about this briefly in an older episode about network effects. In a traditional database, if you want to find out who my brother's housemate's favorite wine is, the computer has to scan a table, find your I-D, jump to a mapping table, find Daniel's I-D, jump to another table, and so on. It is like looking through a giant filing cabinet for every single step.

Herman

Exactly. We call that the join penalty. As your data grows, those joins become exponentially more expensive. But in a graph, it is called index-free adjacency. The node for Corn literally has a physical memory address for the edge leading to Herman. It is like a map where the roads are already paved. You just follow the path. It does not matter if you have ten nodes or ten billion; following a relationship takes the same amount of time. It is O of one complexity for each hop, rather than O log N for a search.

Corn

So if graph databases are so efficient at relationships, why are we seeing this massive surge in vector databases? Daniel mentioned Pinecone and the idea of being vector-native for A-I. How does that fit into the picture?

Herman

This is where the story gets really interesting here in twenty-twenty-six. Vector databases and graph databases solve two completely different problems, but A-I needs both. A vector database is all about semantic similarity. It does not care about the relationship between two things in a factual sense; it cares about how similar they are in meaning based on their mathematical embeddings.

Corn

Give me an example of that.

Herman

Okay, imagine you have a thousand documents about fruit. In a vector database, the word orange and the word tangerine will be very close to each other in a mathematical space because they are semantically similar. When you ask an A-I a question, it uses a vector search to find pieces of text that are related to the topic. It is great for finding the right neighborhood of information.

Corn

But it is not great at facts, is it?

Herman

Precisely. This is the big hallucination problem we have been fighting for years. A vector search might find a document that mentions oranges and a document that mentions white wine, and the A-I might just mash them together and tell you that oranges are made of wine. It understands similarity, but it does not understand the structure of the truth. It lacks the logic of the edge.

Corn

And that is where the graph comes back in.

Herman

Exactly. The trend we are seeing right now—and Daniel hit the nail on the head by looking at Neo-four-j—is the move toward something called Graph-R-A-G. That stands for Retrieval-Augmented Generation using a knowledge graph. Instead of just finding similar chunks of text, the A-I first looks at the graph to see the actual, verified relationships. It sees that Orange is a Fruit and White Wine is an Alcoholic Beverage. It uses the edges to navigate the facts before it starts generating a response.

Corn

Research has shown that Graph-R-A-G significantly improves accuracy on multi-hop questions compared to standard vector retrieval. That is a massive jump.

Herman

It is huge. And a multi-hop question is exactly what a graph is built for. A question like, Which of my housemates who has a cold is drinking something that contains alcohol? That requires jumping from the person, to their health status, to their housemate status, to the liquid in their glass. A vector database struggles with that because it is trying to find one paragraph that contains all those answers. A graph database just walks the edges.

Corn

So, to Daniel's big question: do we think the backbone of storage is going to shift toward these vector-native and graph formats, or will S-Q-L keep its crown?

Herman

I think we are seeing the end of the database monogamy era. For a long time, the advice was just put it in Postgres. And honestly, Postgres is putting up a hell of a fight. Have you looked at the latest updates for p-g-vector?

Corn

I have. It is impressive. They have commoditized the vector part so well that for a lot of mid-sized companies, a specialized vector database like Pinecone is starting to feel like an unnecessary extra cost. If you can keep your users, your transactions, and your embeddings all in one Postgres instance, why wouldn't you?

Herman

Right, but there is a limit. When you get into the hundreds of millions of vectors or incredibly complex relationship webs, the overhead of a relational database starts to show. I think the future is polyglot, but with a twist. We are seeing Neo-four-j add native vector support, and we are seeing vector databases try to add metadata filtering that looks a lot like S-Q-L.

Corn

It is a convergence. Everyone is moving toward the center. But I suspect the primary backbone will depend on the nature of the application. If you are building a social network or a supply chain tracker, your backbone is a graph. If you are building a simple search engine, it is a vector store. If you are doing accounting, it is still S-Q-L.

Herman

I agree, but I would add one thing. I think the knowledge graph is becoming the new gold standard for enterprise A-I. Companies are tired of their chatbots making things up. They want a source of truth that they can audit. You can't really audit a vector space; it is just a bunch of floating-point numbers. But you can audit an edge in a graph. You can look at it and say, Who created this relationship? When was it last verified?

Corn

That auditability is so important. We talk a lot about trust in A-I, and you can't have trust without traceability. If the A-I says Daniel is drinking wine because he has a cold, I want to be able to trace that back to a specific edge in the database that connects those two facts.

Herman

Exactly. And that brings us back to Daniel's question about how this is all going to play out. I think we are going to see a lot more developers reaching for Neo-four-j as their secondary database. They will keep their main records in S-Q-L, but they will sync a portion of that data into a graph specifically to power their A-I agents. It is about giving the A-I a map of the world instead of just a pile of books.

Corn

You know, it is funny you say that. I was reading a paper the other day about the cost of indexing. Building a full knowledge graph from unstructured data is still really expensive. You have to run the text through an L-L-M to extract the entities and the relationships. It is not like S-Q-L where you just dump a C-S-V file and you are done.

Herman

That is the big hurdle. We are in the manual labor phase of graph building. But we are seeing tools now—automated graph extractors—that are getting much better. They can take a thousand P-D-Fs and automatically build a Neo-four-j graph by identifying the recurring nouns and how they connect. Once that becomes cheap and reliable, the game is over for traditional search.

Corn

So, if you are a developer listening to this today, where do you put your time? Do you go deep on Cypher—which is the query language for Neo-four-j—or do you stick with learning how to optimize H-N-S-W indexes in a vector store?

Herman

Honestly? Learn both, but prioritize the logic of the graph. Vectors are becoming a commodity. Every database has them now. Knowing how to tune a vector index is going to be like knowing how to tune a B-tree index in S-Q-L; it is a basic skill. But knowing how to model a domain as a graph? That is where the real architectural value is. It requires you to actually understand the business you are working in.

Corn

It is the difference between being a mechanic and being an urban planner. The mechanic knows how to make the car go fast—that is the vector optimization. But the urban planner knows where the roads should go to actually get people where they need to be.

Herman

I love that analogy. And for Daniel, the urban planner in our house, he is already thinking about the roads. He is just trying to figure out if those roads should be paved with vectors or edges.

Corn

Well, based on his current state, I think his roads are mostly paved with orange peels and wine corks. But let's talk about the practical side for a second. If someone wants to start with Neo-four-j today, what is the biggest mistake they usually make?

Herman

They try to model it like S-Q-L. They create these giant nodes with fifty properties, effectively turning the node into a table row. The whole point of a graph is to break those properties out into their own nodes if they are shared.

Corn

Right. Instead of having a property on a user node that says city equals Jerusalem, you should have a separate node for Jerusalem and an edge that says lives in.

Herman

Exactly! Because then, when you want to find everyone who lives in Jerusalem, the database does not have to scan every user. It just goes to the Jerusalem node and follows all the lives in edges backward. It is instantaneous. That is the power of the edge. It is not just a line on a diagram; it is a bidirectional shortcut.

Corn

It makes me think about how we store our own memories. We don't have a giant spreadsheet in our heads. If I think of the word Jerusalem, I immediately get hits for the smell of the market, the sound of the bells, and the sight of Daniel drinking wine. Those are all edges. My brain is a graph database.

Herman

It totally is. And that is why A-I feels so much more natural when it is backed by a graph. It is mimicking the associative nature of human thought. When we talk about A-I wanting to be vector-native, what we really mean is that A-I wants to understand meaning. But when we say it needs a graph, we mean it needs to understand context. Meaning without context is how you get hallucinations.

Corn

So, to wrap up Daniel's question about whether S-Q-L and vector databases will continue to play separate roles—I think the answer is a soft yes, but they are becoming features of each other. We are moving away from the idea of a vector database as a standalone thing. It is becoming a capability that you expect from your primary data store, whether that is relational or graph-based.

Herman

Right. I don't think Pinecone is going away, especially for massive-scale applications, but I think for the average developer, the choice will be: do I use the vector features in my S-Q-L database, or do I use the vector features in my graph database? And if your data is highly connected, the graph is going to win every time.

Corn

It feels like we are entering the era of the intelligent data layer. The database isn't just a place where you put things and hope to find them later. It is a part of the reasoning engine itself.

Herman

Absolutely. And that is why I get so excited about this stuff. We are finally moving past the era of dumb storage.

Corn

Well, speaking of intelligent things, or at least things that try to be, we should probably check on Daniel and see if his orange-wine experiment has yielded any new scientific breakthroughs.

Herman

Or at least see if he has managed to create a graph of all the napkins he has used this morning.

Corn

Before we sign off, I want to remind everyone that if you are enjoying these deep dives into the weird prompts Daniel sends us, we would really appreciate it if you could leave us a review on your podcast app. Whether it is Spotify or Apple Podcasts, those ratings really help other people find the show.

Herman

It genuinely does make a difference. We see every single one of them, and they keep us motivated to keep digging into these rabbit holes.

Corn

And if you want to reach out or see the archives of our past two hundred and sixty episodes, you can find us at my-weird-prompts-dot-com. We have an R-S-S feed there and a contact form if you have a prompt of your own that you want us to tackle.

Herman

Just maybe don't send them in while you are on a wine and orange juice bender. Or do. We don't judge.

Corn

Speak for yourself, Herman. Anyway, thanks for listening to My Weird Prompts. I am Corn.

Herman

And I am Herman Poppleberry. We will catch you in the next episode.

Corn

Stay curious, and watch out for those edges. They are more important than you think.

Herman

Bye everyone!