Episode Book

EPISODE #5770

Talking to Your Data: What MCP Doesn't Solve

PUBLISHED	8 October 2026 at 08:35 UTC
DURATION	25:03
LISTEN	myweirdprompts.com/episode/talk-to-your-data-mcp/
HOSTS	Corn, Herman
SCRIPT MODEL	deepseek-flash
VOICES	chatterbox-regular (chatterbox-local)
PIPELINE	V5.2
BOOK GENERATED	19 October 2026 at 17:39 UTC

SYNOPSIS

Every AI platform is racing to ship data connectors, but "talk to your data" hides five distinct jobs — and only one of them is the AI writing SQL. This episode traces what actually happens between a question and a valid query: schema ingestion, semantic grounding, query generation, execution with error feedback, and interpretation back to language. We look at why a semantic layer beats raw text-to-SQL by sixteen points on the same model, how agentic loops with actor-critic and execution feedback close the correctness gap, and whether specialized NL-to-SQL models still earn their place against frontier general models. Then we ask the uncomfortable question underneath all the leaderboard numbers: if roughly a third of the reference answers in the standard benchmarks are wrong, what exactly are we measuring?

TRANSCRIPT



Corn

A company named dbt ran a benchmark this year where the same question hit a hundred percent accuracy through a semantic layer and eighty-four percent through raw text-to-SQL. Sixteen points. Not because the model got smarter, but because somebody had written down what the tables meant.



Herman

Which is the whole episode, right there.



Corn

It's the whole episode, and it's the thing every product deck in the industry is currently skating past. Here's what Daniel wrote in this week. He's watching every major AI platform race to ship data connectors, and the phrase "talk to your data" is now in every deck. But he wants the actual implementation, not the marketing. He's noticed everything in agentic AI is standardizing around MCP, and he's noticed that talking to data involves some very specific technical things that the protocol doesn't obviously cover. For SQL, he breaks it into three pieces. The model needs general relational principles. It needs a semantic understanding of what the tables are for, before it can answer even a simple question. And then there's the execution process: natural language in, valid SQL out, results parsed and interpreted, maybe more queries sent, everything translated back to language.



Herman

And then he asks three things.



Corn

Three things. How does the query-forming process actually happen under the hood. Whether specialized natural-language-to-SQL models still play an important role, or whether general models have moved past needing them. And because we can't cover every database with the same answer, how the challenge changes when the target is a document store or a graph store.



Herman

So let's start with what the pipeline actually looks like, because "talk to your data" hides five distinct jobs.



Corn

Name them.



Herman

Schema ingestion first. You have to get the tables, columns and relationships in front of the model. Then semantic grounding, which is inferring what those things mean. Then query generation. Then execution with error feedback. Then interpretation back to natural language. Five stages, and only one of them is the thing people mean when they say "the AI writes SQL."



Corn

Stage two is where the bodies are buried.



Herman

Stage two is where the entire literature is currently buried. But before we get there, we should be fair to MCP, because Daniel's right that that's the standardization story.



Corn

Define the boundary.



Herman

MCP came out of Anthropic in November of twenty twenty-four. It's now governed by the Agentic AI Foundation under the Linux Foundation, which is the tell that nobody thinks this is a vendor feature anymore. OpenAI adopted it in March of twenty twenty-five. Google and Microsoft are on board. There are north of five hundred tool connectors in the ecosystem. On the database side specifically you've got sql-mcp covering eight engines, MariaDB ships an official server, there's go-db-mcp, a project called coremcp for legacy



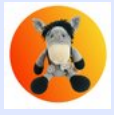
Herman (continued)

MSSQL, and Google's Toolbox for AlloyDB, BigQuery and Cloud SQL.



Corn

And what does it actually standardize?



Herman

Transport and tool discovery. How the agent finds your server, how it lists schemas, how it fires a query, how the result comes back. That's a real achievement. Before MCP every one of those connectors was a bespoke integration.



Corn

What it doesn't do is the interesting part.



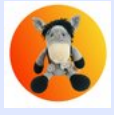
Herman

What it doesn't do is schema semantics, ambiguity resolution, or query correctness. Those stay in the application layer. MCP gives you the pipe. It has no opinion about what's in it.



Corn

So the pipe is solved and the water is still a research problem.



Herman

There's a nice concrete detail on the plumbing too. If you're on ChatGPT Plus or Pro and you attach a custom connector, it's read-only. Write-capable MCP is gated to Business, Enterprise and Edu. So the consumer version of "talk to your data" literally cannot mutate anything.



Corn

Which is the correct product decision and also a useful reminder that this is still being fenced in.



Herman

On the writing side, OpenAI ships a sample repo called MCPKit for secure data connectors, which is the acknowledgment that the security surface is the part people get wrong.



Corn

So MCP gives us the pipe. What flows through it is where the five stages get hard. Start with the mechanism. What actually happens between the question and the SQL?



Herman

dbt Labs put the core tension about as cleanly as anyone has, back in April. The LLM has to infer the semantics of your data from structural clues. Table names, column names, relationships. Then it writes a query from scratch every time. And their line is the one that should worry anybody shipping this: there's no guardrail between the question and the generated SQL.



Corn

No guardrail. That's the sentence.



Herman

So the field's response has been to build the guardrail, and it's not a single translation step. It's an agentic loop. There are five architectural patterns showing up repeatedly, and they're not mutually exclusive.



Corn

Go through them.



Herman

Semantic-layer mediation first, and this is the one I'd flag as most interesting. There's a paper from the middle of this year describing a system that decouples semantic intent from physical SQL execution. The agent doesn't write SQL. It reasons over a curated semantic layer and emits an intermediate Semantic Model Query. Then a deterministic compiler translates that into dialect-specific SQL. Running on Gemini 3 Pro it hits ninety-four point one five percent execution accuracy on Spider2-snow.



Corn

Hold on. The agent doesn't write SQL?



Herman

It writes an intent representation, and a compiler that can't hallucinate turns it into SQL. That's the design move. It's the difference between asking someone to draft a legal contract from memory and asking them to pick clauses from an approved library.



Corn

And ninety-four percent is well above what bare models get on the same family of benchmarks.



Herman

Considerably. Second pattern is multi-agent orchestration. A system called AgentNLQ uses an orchestrator that plans, reflects and self-corrects, plus something they call schema enrichment, which builds context-aware metadata. Seventy-eight point one percent semantic accuracy on BIRD.



Corn

Third.



Herman

Actor-Critic. One model writes the SQL, a second model evaluates it, and they iterate until the Critic approves. That's been around since twenty twenty-four and it shows up everywhere now.



Corn

Fourth.



Herman

Agentic views. AV-SQL decomposes complex queries into agent-generated Common Table Expressions. You build the query in stages instead of trying to emit one monster statement. Seventy point three eight percent on Spider 2.0.



Corn

And fifth is the one that should be most obvious.



Herman

Execution feedback. Nearly every modern system runs the intermediate SQL and refines based on the error. MARS-SQL, SERL-SQL, CoTE-SQL, they all do it. The database itself becomes a reviewer.



Corn

Which is the only component in the loop that can't be talked out of its opinion.



Herman

That's the honest summary of why it works. A Critic model can be flattered into approving bad SQL by a confident Actor. An engine returning a syntax error cannot.



Corn

Which raises what Daniel actually asked. If the loop is this good, do we still need a specialist model at all?



Herman

The evidence cuts both ways, and I want to give you both sides before I say where I land.



Corn

Both sides.



Herman

Against specialists: dbt's benchmark this year shows frontier general models at ninety percent for Sonnet 4.6 and eighty-four point one for GPT-5.3 Codex on their text-to-SQL track. In twenty twenty-three GPT-4 was at thirty-two point seven percent on the same kind of task. dbt's conclusion is blunt: the choice of model matters less than you'd think. And they add a line I like, that the biggest model isn't always the best model for structured data tasks.



Corn

That's a big shift in three years.



Herman

It's a bigger shift than the leaderboards make it look, because the leaderboards measure the hard tail and the industry mostly lives in the easy middle.



Corn

Now the other side.



Herman

Top of the BIRD leaderboard is not a frontier model. GrainSQL out of Purdue sits at eighty-two point nine five on test, DataGallery at eighty-two point three nine. Bare frontier models on BIRD dev are GPT-5.5-xhigh at seventy-two point five five and Claude Opus 4.6 at seventy point one five. So there's roughly a ten-point gap, and that gap is being closed by specialists.



Corn

Ten points is not nothing.



Herman

Ten points on a benchmark is the difference between shipping and not shipping for some applications. And then there's the cost argument, which is where small specialists win. SLM-SQL gets a half-billion parameter model to fifty-six point eight seven BIRD EX and a one-point-five-billion model to sixty-seven point zero eight. There's an agentic small-model system that resolves about sixty-seven percent of queries locally at eight-tenths of a cent per query, versus nine point four cents for LLM-only. That's an eleven-x cost difference.



Corn

Say that ratio again.



Herman

Eight-tenths of a cent versus nine point four cents. And if you're running a customer support tool that gets fifty thousand queries a day, that's a line item.



Corn

Plus the privacy angle.



Herman

Plus privacy. A model running on your own hardware never sends the schema anywhere. GEMMA-SQL gets Gemma 2B to sixty-six point eight percent test-suite accuracy. And LIMIT shows Qwen3-8B reaching sixty-nine point one on BIRD and eighty-eight point nine on Spider EX with only about eight hundred curated training samples. Eight hundred samples. That's an afternoon of data work, not a fine-tuning project.



Corn

So where do you land?



Herman

Specialists are no longer needed for basic SQL knowledge. General models have that, and pretending otherwise is nostalgia. They persist for three reasons. Cost, latency and edge deployment. Squeezing the last ten points on hard schemas. And domain-specific fine-tuning where the vocabulary is unusual. But the framing of the whole debate is wrong.



Corn

How so.



Herman

Because the real competitor to text-to-SQL isn't a specialist model. It's the semantic layer. And I don't think that's widely understood yet.



Corn

You already gave me the number.



Herman

A hundred percent for GPT-5.3 Codex through dbt's Semantic Layer, versus eighty-four point one for the same model doing raw text-to-SQL. Same model. Sixteen points of difference from nothing but the interface.



Corn

And the qualitative difference is bigger than the number.



Herman

That's the part dbt nails. Their line is that with text-to-SQL, failure looks like a plausible but incorrect answer. With the Semantic Layer, failure looks like an error message. And they add: for anything going to a board deck, an auditor, or a company KPI dashboard, that difference is everything.



Corn

Because a wrong number that looks right is worse than no number.



Herman

A wrong number that looks right is the single most expensive artifact in enterprise software. A model that says "I can't answer that" gets escalated to a human in thirty seconds. A model that confidently returns last quarter's revenue with the wrong join in it gets emailed to the board.



Corn

Which reframes the whole thing. The competitor isn't the specialist model. It's the semantic layer.



Herman

And it gets bigger when you leave SQL behind.



Corn

Before we do, there's a problem underneath all these numbers.



Herman

The benchmark integrity question.



Corn

Which undercuts everything you just cited.



Herman

It does, and it has to be said. There's a paper called SALUS that automated an audit of NL-to-SQL benchmarks and estimated annotation error rates around thirty-seven percent on BIRD and twenty-seven percent on Spider.



Corn

Thirty-seven percent of the questions have wrong answers?



Herman

Thirty-seven percent of the reference answers are estimated to be wrong or ambiguous. Which means some portion of what the leaderboard is measuring is agreement with a flawed key. Human performance on BIRD test is ninety-two point nine six. If the annotation error estimate is anywhere near right, that gap between GrainSQL and humans is either smaller than it looks or larger than it looks, and we don't know which.



Corn

We're grading on a ruler with known defects.



Herman

And everybody's optimizing against it anyway, because it's the only ruler anyone agrees on.



Corn

And that pushes us off SQL entirely. Which is where Daniel's third question lives.



Herman

Document databases first, because the naive assumption is that it's a dialect swap. It isn't.



Corn

Explain the difference.



Herman

MongoDB doesn't have tables. It has collections of documents with nested structure, and the query language is a procedural aggregation pipeline. You're not declaring what you want, you're assembling a sequence of stages that transform data. That's a different cognitive shape from SQL, and it turns out it's a different shape for models too.



Corn

Evidence?



Herman

There's a benchmark called TEND, the first MongoDB-native text-to-NoSQL benchmark. One thousand two hundred and ten tasks across eleven databases. Their finding is one sentence that should make anyone planning a port nervous: LLMs with strong NL2SQL performance degrade substantially on TEND.



Corn

Same models, same task family, different query language, big drop.



Herman

Big drop. And there's a benchmark from this month called AptMQL-Bench that tested the obvious shortcut, which is taking SQL pipelines and mechanically converting them.



Corn

The shortcut doesn't work.



Herman

Naive SQL-to-MQL conversion fails to migrate six of twenty-one BIRD databases outright, and where it does run, it silently drops up to twenty-five point nine percent of rows.



Corn

Silently.



Herman

Silently. That's the word that matters. It doesn't error. It returns an answer that's missing a quarter of the data, and the aggregate on top of it is wrong, and nobody knows.



Corn

That's the worst possible failure mode. Wrong and confident and no exception thrown.



Herman

On raw accuracy, Claude Opus 4.5 gets fifty-seven point three eight on text-to-MQL without external knowledge evidence, seventy point three four with it. Those are low numbers compared to what the same model does on SQL.



Corn

Which brings the specialist argument back.



Herman

It does. EvoMQL reports seventy-six point six in-distribution and eighty-three point one out-of-distribution execution accuracy on natural-language-to-MongoDB with a three-billion-parameter model. Three billion. That beats the general model substantially, which is the opposite of the SQL story.



Corn

SQL, general models have caught up. MQL, they haven't.



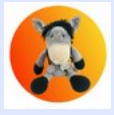
Herman

And there's a knock-on effect hiding in that research that I think is the most interesting thing in the whole document set. Document schemas should be designed from access patterns, not mirrored from the relational foreign-key graph.



Corn

Repeat that, because it's a real claim.



Herman

If you're building a document store specifically so people can query it in natural language, the right way to design the schema is to start from the questions people ask. Not to take your relational model, decompose it into collections, and hope. The database design itself has to change.



Corn

So it isn't a model problem at all at that point.



Herman

It's a data-modeling problem. The model is downstream of a decision made by an engineer two years earlier about how to nest things.



Corn

Graph databases next.



Herman

Least mature by a distance. There's a twenty twenty-four survey that states it plainly: while research on LLM-driven query generation for SQL exists, similar systems for graph databases remain underdeveloped. In their evaluation Claude Sonnet 3.5 outperformed GPT-4o, Gemini Pro 1.5 and Llama 3.1 8B on Cypher generation, which tells you how early we are, because that was not a landslide.



Corn

What's the distinctive failure pattern?



Herman

For SPARQL it's URI hallucination. Models invent identifiers. Graph databases are addressed by exact IRIs, so if the model makes one up you get a query that's syntactically perfect and returns nothing, or returns the wrong subgraph. PGMR solves it in a way I find elegant.



Corn

Go on.



Herman

The LLM emits a placeholder instead of an identifier. Then a non-parametric memory module resolves the placeholder against the real URI space. The result is described as near-complete suppression of URI hallucinations.



Corn

So the fix isn't a better LLM.



Herman

The fix is not asking the LLM to remember something it has no reliable way to remember. It's the same insight as the semantic-layer mediation we talked about. Take the part the model is bad at and give it to a component that can't fail.



Corn

And MCP is showing up here too.



Herman

There's work called Agentic SPARQL that evaluates SPARQL-MCP-powered agents on federated knowledge-graph question answering. And the open problem, still on the roadmap rather than solved, is queries that span multiple graphs. Text2Cypher across a federation is where the field ends and the research begins.



Corn

So let's put the three side by side.



Herman

SQL is largely solved for basic fluency, and the remaining gap is schema understanding. MongoDB is partially solved, and the specialist argument is real there because general models aren't close yet. Cypher and SPARQL are barely solved, with no dominant approach and fewer benchmarks. And the reason is SQL-centricity.



Corn

Decades of training data.



Herman

Decades of benchmarks, decades of Stack Overflow, decades of textbooks, all in one query language. Everything else is a couple of years old in comparison. And the naive conversion path across that gap actively corrupts data, twenty-five point nine percent of rows.



Corn

Which leaves the gap between the benchmarks and production.



Herman

A commenter going by efromvt on Hacker News in July said it better than any paper. He said with a loop you can get extremely high results on a clean database with a clear question, he's usually seeing high nineties accuracy. And then a messy or ambiguous schema degrades it, and before context engineering he sees rates closer to twenty to thirty percent.



Corn

High nineties. Twenty to thirty.



Herman

Same technique, same model. The only variable is the schema. And the average schema in BIRD has around seven tables. Enterprise schemas routinely run one hundred to five hundred, with column names nobody outside the company could interpret.



Corn

So the person reporting ninety-five percent is testing on something that was built to be understood.



Herman

Built to be understood, and documented in the benchmark itself. The production case is a schema that accreted over a decade and whose documentation lives in four people's memories.



Corn

Which is where every paper in this pile converges.



Herman

There's a line from a database-context-compression paper this year that I'd put on a poster: the main bottleneck is no longer reasoning, but database representation.



Corn

And Tk-Boost is the same claim from a different direction.



Herman

They argue agents fail from misconceptions about the data, and they inject what they call tribal knowledge. Knowledge about column intent that accumulates through experience and never gets written down anywhere machine-readable.



Corn

I watched a number be wrong once because nobody wrote down what a column meant.



Herman

Everyone has.



Corn

Not in a benchmark though. In a real report, seen by people who made decisions with it.



Herman

He's right. There's this one column that had been repurposed years earlier. It still said revenue. It hadn't meant revenue in a long time.



Herman

And the model didn't hallucinate.



Corn

The model didn't hallucinate. It read the column name and believed it. Which is what the column was asking for.



Herman

And that's not an accuracy problem you can fix with a better model. Every model anyone is going to ship in the next five years is going to read that column name and believe it.



Corn

The model isn't wrong. The schema is lying.



Herman

Right, and the fix isn't a model at all. It's somebody writing down what the column means, in a place the model can see it.



Corn

Who does that writing-down? I have an opinion and it isn't the data team.



Herman

The data team owns the pipeline. The person who knows the column changed meaning is the analyst who stopped using it, or the person in finance who remembers the migration. That knowledge doesn't live where the tooling looks.



Corn

It never does.



Herman

Which means the tribal-knowledge injection those papers are doing is really an organizational exercise wearing a technical costume.



Corn

Somebody has to finally write it down. That's the whole fix. It's been the whole fix.



Herman

And the papers can describe the mechanism, but they can't describe what it's like to sit in a room where nobody remembers why the field is named that.



Corn

So the answer to Daniel's question about specialist models is no, not for SQL, and yes, for basically everything else, and neither of those is the interesting part.



Herman

The interesting part is that the field has been arguing about the wrong layer for two years. We've been measuring model capability while the actual constraint has been metadata quality the entire time.



Corn

The benchmark gap between a specialist and a general model is ten points on a ruler with a thirty-seven percent error rate. The gap between a clean schema and a messy one is sixty points.



Herman

Sixty points, measured by somebody who just runs these systems for a living. That's not a research finding, it's a production report, and it's more useful than either leaderboard.



Corn

So if the bottleneck is metadata and not model capability, what happens to the MCP story?



Herman

MCP solved the transport layer beautifully and left the hard part completely untouched. Five hundred connectors, all moving schemas around, none of them explaining what a column means.



Corn

Does the next standardization fight happen over semantic layers and schema documentation formats?



Herman

Or does everybody build their own semantic layer, badly, and re-derive sixteen points of accuracy loss per company.



Corn

And underneath that, the benchmark problem.



Herman

If BIRD has roughly thirty-seven percent annotation error and Spider twenty-seven, then the leaderboard gap between specialists and general models might be smaller than it looks. Or larger. We don't know which, and we're making deployment decisions on the basis of it anyway.



Corn

Which means the field might be optimizing against a ruler with known defects.



Herman

It is optimizing against a ruler with known defects, because it's the only ruler anyone agrees on, and the alternative is no comparison at all.



Corn

And the SQL-centricity.



Herman

Text-to-MQL and text-to-Cypher are years behind, and the naive conversion path drops a quarter of the rows while reporting success. So the industry either waits for those benchmarks to mature, or keeps shipping SQL-shaped solutions into non-SQL problems and calls the row loss a rounding error.



Corn

Which is the decision nobody wants to make in public.



Herman

It's the decision everybody is making in private.



Corn

If you want this without the marketing layer, we're at my weird prompts dot com, and the feed is in the show notes.



Herman

Thanks as always to our producer, Hilbert Flumingtop.



Corn

This has been My Weird Prompts.



Herman

We'll be back soon.