Episode Book

EPISODE #6429

Why Postgres Needs an Analytical Partner

PUBLISHED	3 November 2026 at 19:45 UTC
DURATION	25:51
LISTEN	myweirdprompts.com/episode/postgres-analytical-database-pairings-44769/
HOSTS	Corn, Herman, Clive, Hilbert
SCRIPT MODEL	z-ai/glm-5.3
VOICES	chatterbox-regular (chatterbox-local)
PIPELINE	V5.26
SOURCES	17 cited by the research (listed at the end)
BOOK GENERATED	4 November 2026 at 09:31 UTC

SYNOPSIS

Production applications hammer operational row stores, but scanning history demands columnar engines. Exploring how database pairings evolve across startups, mid-market companies, and enterprises, this breakdown covers the escalation ladder from basic read replicas and embedded tools like pg_duckdb to dedicated platforms like ClickHouse, BigQuery, and Snowflake. It unpacks the true costs of maintaining analytical pipelines, cold-start query queueing, and why larger organizations end up running multiple specialized analytical systems simultaneously.

TRANSCRIPT



Corn

Two databases. That's the whole landscape, apparently, and Daniel wants us to walk it before we get anywhere near the plumbing.



Herman

And I've been looking forward to this one, because it's one of those things where everybody in the trade knows the shape and almost nobody can say when it started being the shape.



Corn

Daniel's framing it as part one of two. He's noticed that most serious data setups end up with an operational database, the thing the application hammers all day, sitting next to some big analytical engine that exists to scan history. Postgres on one side, BigQuery or Snowflake or ClickHouse or Databricks on the other. He wants the actual pairings — what a startup runs, what a mid-sized company runs, what an enterprise runs — and what makes a team pick one partner over the other. Cloud vendor, cost model, real-time versus batch, managed versus self-hosted. And he wants us to place the words warehouse, lake and lakehouse into that picture.



Herman

Next episode is the pipeline between them. Today is just: what are the two boxes, and why is one of them almost always Postgres.



Corn

Before we argue about which second database to buy, we should be precise about why there's a second database at all. The core asymmetry: your operational store is row-oriented, tuned for many small reads and writes all day. You fetch one user, you update one order. The analytical engine is columnar, tuned for scanning enormous stretches of history to answer one question. Those are physically different jobs.



Herman

That difference is why the two-box pattern keeps reappearing no matter how good Postgres gets.



Corn

One honesty flag before we go further, because I'd rather say it now than bury it. There is no authoritative survey of pairings. Nobody publishes "this percentage of companies run Postgres plus BigQuery." ETR measures vendor spending momentum, IDC's market-share report is paywalled, so what we actually have is architectural guidance from vendors and practitioners. Which is real evidence, but it's not a census, and I want to be straight about that.



Herman

That's fair. The pattern is legible from what people build and write about, not from a headcount. Okay, let's walk the three company sizes in order, because the pattern changes completely between them.



Corn

Start at the bottom, because the bottom is my favorite part.



Herman

The startup answer is: no second database at all. Plenty of companies between zero and ten million in annual recurring revenue run their entire analytics surface on the production database. That's not a hack, that's the default state of the world.



Corn

It's also the state I'd defend longest, which surprises nobody.



Herman

It surprises nobody who's ever watched you resist an upgrade. The interesting part is the escalation ladder when the dashboard starts slowing down, because the first rungs are free. Step one, read the query plan. A large share of "we need a warehouse" conversations end at EXPLAIN ANALYZE — somebody finds a sequential scan over a join that was never meant to exist.



Corn

The dashboard is slow because the query is stupid, not because the data is big.



Herman

In a lot of cases, yes. Step two, move reporting onto a read replica, which in most cases buys you another six to twelve months. Step three, pre-aggregate with materialized views, so the expensive counting happens once at night instead of every time somebody opens the dashboard. Step four, add columnar storage inside Postgres itself. Only step five is buy a warehouse.



Corn

And the diagnostic buried in that ladder is the one I'd want people to take away: slow dashboards are usually a storage-model problem, not a data-volume problem. You hit a wall because you're scanning rows to answer a columnar question, not because you have too much data.



Herman

There's a number attached to that, from the same piece of guidance: below roughly five terabytes of hot analytical data, a single well-configured machine usually beats a distributed cluster on both cost and latency.



Corn

Five terabytes. Below that, one big box wins.



Herman

Except here's the tension, and I love this one: ClickHouse's own material says if you try stretching Postgres to handle terabytes of analytical queries, you'll hit a physical wall in column scanning and I/O.



Corn

Same number, opposite conclusion.



Herman

Same terabytes, opposite conclusion. And the difference between those two positions is whether you count the pipeline. Once you move analytics out, you own an ETL process forever, you own a second source of truth, and you own a freshness lag between the two. reptile.haus puts it well: the real cost of a warehouse is not the licence, it's the permanent ETL pipeline, the second source of truth, and the freshness lag that comes with both.



Corn

A warehouse isn't a purchase, it's a mortgage.



Herman

That's actually the best one-line summary of that whole argument I've heard. So between "one big box" and "buy a warehouse" there's now a middle ground that barely existed a couple of years ago, and it deserves its own minute, because it changed the startup ladder. There are now four architectures for doing analytics without leaving Postgres.



Corn

This is the part you've been waiting to get to. I could tell.



Herman

I have been slightly vibrating about it, yes. Architecture one, an engine embedded inside each Postgres backend — pg_duckdb is the big one, DuckDB running inside Postgres, and it hit version one with over a million downloads. Also pg_mooncake, which keeps a



Herman (continued)

columnar mirror of your tables in Iceberg format and pushed Postgres into the top ten of ClickBench doing it. Architecture two, a sidecar on the same host, a separate DuckDB process reading Postgres data — that's pg_lake. Architecture three, columnar storage inside Postgres proper — TimescaleDB's Hypercore, Citus columnar, AlloyDB's columnar engine. Architecture four, query pushdown — Postgres parses the query and ships the heavy part to a separate ClickHouse, that's pg_clickhouse.



Corn

And the interesting numbers are on the pushdown side.



Herman

They are, because the obvious worry with all of this is you're running analytical work on or near the box that's serving your application. ClickHouse's own tests found pg_clickhouse changed OLTP throughput by less than two percent under analytical load, against drops of eighteen to fifty-eight percent for the options that run on the Postgres host itself. So the pushdown pattern mostly protects the operational side.



Corn

Mostly.



Herman

Mostly, because the same benchmark has the asterisk. On TPC-H at scale factor one, a fully pushed-down query drops from about four and a half seconds to fifty-nine milliseconds. Another drops from two seconds to thirty-seven. But the queries that only partially push down, where some fragment has to run in Postgres anyway, come out slower than plain Postgres with no extension at all.



Corn

The middle ground is real but lumpy. You don't find out which kind of query you have until it's slow.



Herman

That's a fair reading, yes. And I'd add the general caveat that these are the vendor's own numbers about the vendor's own extension, so hold them with the appropriate grip. Still — four architectures, a million downloads on the embedded one, that's a real shift, not a blog post.



Corn

It's also the concrete follow-up to something we said back on the episode about why "just use Postgres" isn't always enough. We argued the boundary was fuzzier than the slogan implies, and this is the boundary getting fuzzier on a schedule.



Herman

Right, the slogan keeps being half-right for longer every year. Okay, mid-sized company. The default pattern here has a name in the guidance: the speed layer. Keep the warehouse for data science, batch processing, long-term compliance — the slow, careful, governed stuff. Then replicate a handful of high-value, query-heavy datasets into a real-time engine, and point dashboards and API endpoints at that.



Corn

Postgres stays the operational store, the warehouse stays the record of history, and the columnar engine sits in front as a serving layer.



Herman

Streams can even go straight into ClickHouse for telemetry, bypassing the warehouse entirely. And the trigger for building this is concrete, not vibes: you want sub-second response at the ninety-fifth percentile on fresh data, and you have fifty to a hundred or more concurrent users.



Corn

Why can't the warehouse just do that? It's got the data.



Herman

Two mechanisms, and they're boring in the best way. Queueing, first — many simultaneous queries waiting on shared resources. Second, virtual-warehouse startup overhead. Snowflake-style warehouses auto-suspend when idle to save you money, and when a query arrives after a suspension, there's a cold start before anything runs. Queueing plus cold start, and your p95 latency is measured in spikes. As the data grows, ClickHouse's numbers put the cost-performance gap at twenty-three times or more versus a purpose-built real-time engine.



Corn

The warehouse is built to answer anything eventually, not to answer this question now.



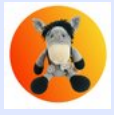
Herman

That's a nice compression of it. So the mid-market stack is three things: operational Postgres, a governed warehouse, and a serving layer in front — and that's not fashion, it's concurrency and cold-start economics.



Corn

Then the enterprise, which is where the pairings stop being a choice and start being a portfolio.



Herman

ETR's cloud data warehouse survey — three hundred twenty-one IT decision makers, most of them at large companies — ranks spending momentum like this: Snowflake at a seventy-two percent Net Score, Databricks sixty-seven, Azure Synapse fifty-seven, BigQuery fifty-six, Redshift fifty-six. Then the floor drops out. IBM at four. OpenText twelve, Oracle fourteen.



Corn

A cliff after the top five.



Herman

A cliff. What I find more interesting than the momentum numbers is how tightly the expected three-year ROI clusters: BigQuery seventy-six percent, Synapse seventy-five, Redshift seventy-four, Databricks seventy-four, Snowflake seventy-three. Nobody can tell these apart on expected return. The differentiation is elsewhere — Snowflake rated most innovative, Microsoft the vendor most respondents would choose if rebuilding from scratch. And Databricks and Redshift each gained eight points year on year while OpenText and Cloudera fell twenty-four and twenty-one.



Corn

So the enterprise answer to "which analytical engine" is mostly "yes."



Herman

It's a governed warehouse for the BI and compliance crowd plus a lakehouse for the data science and ML crowd, and increasingly both, with Iceberg as the shared storage layer underneath. The large enterprise doesn't pick a partner. It acquires them.



Corn

Two of them. Plus the serving layer from the mid-market pattern, if they've got a product with a live dashboard in it.



Herman

Nobody said the org chart was cheap. And now, our regular correspondent rings in from somewhere far away. Clive, what have you got for us?

C

Clive

One-Star Getaways. Tallinn, the Hotel Merikild, one star. I paid for a sea view and got a framed painting of the sea, bolted to the wall, facing the wrong way.

C

Clive

The brochure promised a harbour panorama and a complimentary morning pastry. The pastry was one single rye cracker, and the panorama was the car park of a rival hotel. So if you fancy a disappointing trip this winter, I recommend it.



Corn

A painting of the sea, bolted on. That's the most honest metaphor for a stale dashboard I've ever heard.



Herman

Please don't book it.



Corn

I'm not booking it. I'm saying the rye cracker has the same energy as a free tier.



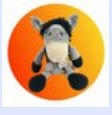
Herman

Moving on. So that's the landscape — the three sizes. The interesting part now is what actually makes a team pick one analytical partner over another, and the first force is the least romantic one: cloud vendor gravity. BigQuery if you're on GCP, Redshift if you're on AWS, Synapse if you're on Azure.



Corn

The pairing is decided by the cloud bill, not by a benchmark.



Herman

In a lot of companies, yes. The hyperscalers' warehouses integrate with the rest of their suite — the data already there, the governance already there, the security already there. Choosing BigQuery when your data lives in GCP isn't laziness, it's not multiplying seams.



Corn

Force two, then. Cost. And I know you've been waiting for the horror stories.



Herman

Because they're the only part of a pricing page anyone remembers. BigQuery bills per terabyte scanned, six dollars twenty-five, first terabyte free each month — or by slots, where a five-hundred-slot Enterprise reservation runs roughly twenty-two thousand dollars a month. The horror story: an unfiltered SELECT star over an eight-hundred-gigabyte partitioned table costs five dollars per run. A dashboard with seventeen tiles, each one triggering that query. That's four and a half thousand dollars a month for one dashboard.



Corn

Seventeen tiles. Nobody multiplied it by thirty.



Herman

Snowflake's version: a forgotten X-Large warehouse left running over a weekend is about two thousand four hundred dollars on Standard. The default auto-suspend is ten minutes, and somebody changed it or somebody's session kept it warm all weekend.



Corn

ClickHouse in the middle of those two, quietly.



Herman

Usage-based, about twenty-five dollars a terabyte a month for storage, and compute from roughly twenty-two cents to thirty-nine cents per unit per hour depending on tier. The structural difference isn't the numbers, it's what triggers them — BigQuery charges you for what you scan, Snowflake for how long you compute, ClickHouse for what's provisioned.



Corn

Which means the cost argument is really a workload-shape argument wearing a pricing hat. Scan-heavy and irregular, on-demand BigQuery punishes you. Spiky and bursty, Snowflake punishes you. Steady and concurrent, that's ClickHouse's home turf.



Herman

Force three: openness and lock-in. Snowflake and BigQuery are closed platforms with proprietary storage formats — Snowflake's micro-partitions, BigQuery's Capacitor. One analyst calls it an inescapable walled garden. Databricks uses open table formats but you can't self-host the platform. ClickHouse Cloud is built on the open-source engine, so you can run the same thing on a laptop or on your own metal.



Corn

The escape hatch, though, is the one Daniel will enjoy, because it's a format story, not an engine story.



Herman

Iceberg. The argument goes that twenty twenty-four was its VHS moment — Databricks and Snowflake both capitulating to the open table format — and that this weakens warehouse lock-in at the storage layer specifically. Your history sits in open Parquet files with Iceberg metadata, and engines become replaceable readers rather than vaults.



Corn

The data outlives the vendor, in other words, which is the only retirement plan I trust.



Herman

Force four: real-time versus batch, the workload shape. ClickHouse's niche is latency-sensitive work — the names people cite are Ramp, GitLab, Statsig, PostHog, Braze, often as a ten-to-twenty-times cheaper replacement for Elasticsearch or Datadog when what you were really doing was analytics on logs.



Corn

And when does the warehouse become mandatory rather than optional?



Herman

Two conditions, both practical. You must join product data to Stripe, HubSpot, Salesforce, Segment, Intercom — many sources, one query, none of which you can pull into your operational Postgres. Or you need append-only history — the record of what was true, not just what is true — and a production database overwrites by design.



Corn

Now the vocabulary, because I want to hold the three words up against the pairings and see if they explain anything.



Herman

They do, and that's the part I find satisfying. A data warehouse is an architecture pattern, not a technology — pick an engine, load consolidated historical data into it, query it. A data lake is raw data in its native format, usually files on object storage, structured and unstructured. A lakehouse is the hybrid: lake-style ingestion of anything, plus warehouse-style transactions and enforced quality. That came out of a Databricks paper in twenty twenty, and the mechanics are open table formats — Delta Lake, Iceberg, Hudi — layered over Parquet, adding transactional semantics, schema enforcement and time travel, so multiple engines can read and write the same tables.



Corn

Map them onto the pairings.



Herman

Postgres plus BigQuery or Snowflake is the classic warehouse pairing: operational store feeding a consolidated, governed analytical store. Postgres plus Databricks is the lakehouse pairing: the analytical side is an open format over object storage, and the warehouse is a query engine sitting on top. Postgres plus ClickHouse is neither, strictly — it's a serving-layer pairing. The analytical engine is a query surface, not the system of record for history. The warehouse, or lake, remains the record.



Corn

So the same engine can play different roles depending on where the history lives.



Herman

And the misconception that costs people: thinking warehouse, lake and lakehouse are three competing products you choose between at the store. They're patterns, and one company can legitimately run all three at once. Inside the lakehouse you often find the medallion architecture — bronze for raw, silver for cleaned, gold for aggregated.



Corn

Which sounds like a product and is actually just a naming convention for layering.



Herman

Just a naming convention. Somebody should trademark it anyway, somebody will.



Corn

Two things I want to put on the table before we run out of room. First, the vendors argue with themselves about whether you need two systems at all.



Herman

ClickHouse does, specifically. Their top-five-warehouses piece calls the dual-system Modern Data Stack a pain — data duplication, inconsistent definitions, spiraling total cost of ownership. Their real-time analytics guide recommends exactly that dual-system speed layer. Both are their own publications.



Corn

That's not a gotcha, though. It's an honest sign the answer depends on which workload you're optimizing for.



Herman

I agree. The second thing?



Corn

The one that quietly rewrites the whole ladder we started with. Text-to-SQL agents. An agent on top of your data generates unpredictable, unoptimized queries at machine frequency — nobody EXPLAIN ANALYZEs a query an agent wrote at two in the morning.



Herman

And that's precisely the workload that makes the concurrency and blast-radius arguments for a separate analytical system stronger. The ladder that says "you probably don't need a warehouse" may be quietly obsolete for any team that put an agent on top of its data.



Corn

The agent doesn't read the dashboard. It just wants the answer fast, forever, and it never gets tired of asking.



Herman

"Fresh." That's the word you two keep using, and I'd want it pinned down. I run a small pair of my own. Operational side is Postgres, few hundred megs, nothing. Analytical side takes the nightly copy. And I keep a reconciliation sheet.



Corn

A what?



Hilbert

A spreadsheet. Printed, actually, in a folder. Counts I check against both sides, every Monday. Eleven twenty in the morning, after the copy finishes. If the analytical side disagrees with the operational side, the operational side wins. That's just the rule. It's the one writing the numbers as they happen; the other one is telling you a story about yesterday.



Herman

Printed.



Hilbert

Herman, you explained the pipeline to us for a good while earlier. It was fine. But nobody said who checks it. The copy doesn't check itself. And your "fresh" — you both said "fresh data" like it's one thing. Anything older than about forty seconds I don't consider fresh for a count someone might act on. Most of these serving layers are proud of a minute or two. That's a long time if a number is moving.



Corn

Forty seconds.



Hilbert

On the Mondays. There was a bloke in the next office, Rex, who could reconcile his figures in four minutes flat, every time, and made sure everyone knew. I never got under six. That's in the folder too.



Herman

The folder's the part I keep coming back to. You built a manual reconciliation process for a two-database setup one man runs.



Hilbert

I did the dashboard once, deliberately, wrong. Kept a stock figure a day behind for a fortnight on purpose, because the correct number would have triggered an order nobody needed yet. That's a legitimate way to run a small system and I'd do it again. The sheet's in the folder with the wiring diagram. The database itself I threw out years ago. Paper keeps better.



Corn

Rex and his four minutes is the grudge that outlived the hardware.



Hilbert

Rex was quick, I'll give him that. He was never right any faster than me, though. Anyway. Whoever's actually keeping a pair like that running, their Monday looks like mine. It's not exciting. It just has to be done by someone who notices when it stops agreeing.



Corn

He has a point about "fresh" being a number, not a vibe. We used it about six times and never once said how many seconds we meant.



Herman

And his rule — the operational side wins a disagreement — is exactly the serving-layer logic from the mid-market pattern, arrived at from a printed folder. The record of what's happening outranks the copy of what happened. Though I'd want the reconciliation automated, not Monday-morning-by-hand.



Corn

Forty seconds is stricter than most vendors' definition of real time, and he holds a dashboard to it with a folder.



Herman

Which tells you the freshness lag isn't a technical parameter. It's a tolerance, and it's set by whoever has to act on the number.



Corn

Cutting-room floor before we close. The bit I couldn't fit: there's an experiment called pgrust, not production ready, that got summing five hundred million numbers down from about twenty seconds to a hundred and thirty-five milliseconds inside Postgres — a claimed three-hundred-fold gain on ClickBench. The Postgres-native middle ground isn't



Corn (continued)

just extensions that exist, it's raw speed still on the table.



Herman

That's the trend line worth watching. The gap between "one big box" and "the wall" keeps moving.



Corn

So, the open question for next time. If the two-box pattern is real and the vocabulary makes the pairings legible, the interesting question is no longer which two boxes — it's what moves between them, how often, and who owns the pipeline when it breaks. Because the five-terabyte line cuts both ways depending on whether you count the pipeline's permanent cost: the ETL, the second source of truth, the freshness lag. That's part two.



Herman

And Hilbert's forty seconds is as good a measure of that cost as any benchmark I've seen.



Corn

Thanks to Hilbert Flumingtop for producing this one, as ever. If you want more of this, try episode twelve thirty-three, Why Just Use Postgres Isn't Always Enough; episode eleven, How Does Fine Tuning Work Anyway; and episode three, Safetensors or something else. This has been My Weird Prompts. If you've enjoyed it, the website is my weird prompts dot com, and a review wherever you listen helps more than you'd think.



Herman

We'll be back soon. See you tomorrow.

SOURCES

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

1. **Google Cloud [primary]**
   https://docs.cloud.google.com/bigquery/docs/best-practices-costs

2. **ClickHouse, 2026 [primary/vendor]**
   https://clickhouse.com/resources/engineering/how-to-choose-a-database-for-real-time-analytics-in-2026

3. **ClickHouse, 2026 [primary/vendor]**
   https://clickhouse.com/resources/engineering/top-5-cloud-data-warehouses

4. **ClickHouse, data as of 2026-10-29 [primary/vendor]**
   https://clickhouse.com/resources/engineering/best-postgres-extensions-for-analytics-2026

5. **ClickHouse, 2026 [primary/vendor]**
   https://clickhouse.com/resources/engineering/clickhouse-vs-postgresql-analytics

6. **Sacra, 2026-02-25**
   https://sacra.com/research/clickhouse-160m-arr-growing-256-yoy/

7. **PitchBook (snippet only; full fetch not retrieved)**
   https://pitchbook.com/news/articles/big-data-wars-clickhouse-has-a-playbook-to-beat-snowflake-at-its-own-game

8. **ETR, 2025-08-27**
   https://research.etr.ai/blog-observatory/pure-plays-and-cloud-service-providers-dominate-cloud-data-warehouse-market

9. **IDC, Sept 2026 (abstract only)**
   https://my.idc.com/getdoc.jsp?containerId=US53730226

10. **reptile.haus, 2026**
    https://reptile.haus/journal/you-probably-dont-need-data-warehouse-postgres-analytics-2026/

11. **Max Musing, Basedash, 2026-05-14**
    https://www.basedash.com/blog/when-to-add-a-data-warehouse-signals-your-startup-has-outgrown-its-production-database

12. **Valiotti Data, 2026**
    https://valiotti.com/bigquery-vs-snowflake-2026/

13. **last edited 2026-10-03**
    https://en.wikipedia.org/wiki/Lakehouse

14. **last edited 2026-09-06**
    https://en.wikipedia.org/wiki/Data_lake

15. **last edited 2026-08-17**
    https://en.wikipedia.org/wiki/Data_warehouse

16. **WeBridge, 2026**
    https://webridge.co/tech-stack/data-analytics-tech-stack

17. **HN Who is hiring (Oct 2026), Yogi: python (fastapi, prefect), postgres/clickhouse**
    https://news.ycombinator.com/item?id=49935441