EDB POSTGRES AI

Converged Analytics: The End of the Hidden Tax on Every Decision

How data movement drains cost, control, and AI readiness—and the architecture that ends it

Dunith Danushka
Principal Technical Field PMM

Acknowledgments

Author
Dunith Danushka

Co-authors
Purnima Phansalkar
Dave Stone

Peer Review
Chris Chiappone
Jack Christie
Keith Darter

Structure and Content
Dunith Danushka
Dave Stone

Design & Editorial
Ben Alamilla
Maya Fisher
Mary VanClay

Contents

Executive summary 4

History of analytics 5

How data analytics became fragmented, and why that matters now 5

The constant 5

In the beginning, one system did everything 5

The data warehouse and the OLTP/OLAP split 5

An explosion of specialized systems 6

HTAP: The first attempt to converge 6

Best of breed and the swing to decentralization 7

The sprawl we live in now 7

Problems of a modern business 8

Why the old trade-off finally breaks 8

Cost: The tax you never see on an invoice 8

Engine and data sprawl 8

Predictability: The bill that punishes you for winning 9

Control: Who really owns your data 9

AI: The reckoning 10

Why patching won't work 10

The use case you need to solve for 11

The decision legacy and the cloud both get wrong 11

Fresh + complete + trusted + governed 12

Why legacy can't do it 12

Why the cloud can't either 12

Now remember who's deciding 13

This is the use case you need to solve for 13

So where do we turn? 14

What we've been building—and the companies who bet on it 14

What we built 16

Converged analytics, in practice 16

Start with a foundation that's already proven 16

One shared source of truth, in open formats 16

PGAA: The part that makes it converge 17

The right engine for each job 17

Governed once, owned everywhere 18

Governance enforced at the data, not bolted on around it 18

Now look back at that payment decision 18

What the numbers say 19

An independent benchmark, and why these metrics matter for converged analytics 19

Cost you can actually plan around 19

Performance that holds when everyone shows up at once 19

And the honest part 20

Where this goes next 21

Appendix: How it works 22

A.1. A reference architecture 22

A.2. The components 23

A.3. Consistency model 24

A.4. Deployment 24

Executive summary

Analytics has spent 50 years becoming more capable and more fragmented at the same time. To serve different jobs, businesses assembled a stack of specialized systems: a transactional database, a warehouse, a data lake, a real-time engine. Each was wired to the others with pipelines. The result is an environment in which most of the cost, delay, and risk lives not inside any one system but in the constant movement of data between them. Put plainly: Most organizations don't have an analytics problem. They have a data-movement problem.

That was a manageable trade-off when "analytics" meant a report read the next morning. But today's real-time decisioning and AI agents need current, complete, and trustworthy context at the moment they act, and every extra copy and pipeline between the question and the answer increases data vulnerability. Moving data is also a loss of control. Every copy is another place data can sit in the wrong jurisdiction, be reached by the wrong hands, or fall outside a clean audit trail. Sovereign control of data—where it lives, who can touch it, and whether every action is provable after the fact—has become a condition of doing business.

The industry has tried to fix this before. HTAP (hybrid transactional/analytical processing) promised a single engine for both transactions and analytics; it stalled because no single engine is good at everything. The more durable answer is convergence: Keep the right engine for each job, but run them all over one shared, governed copy of the data instead of copying that data from system to system.

This paper traces how analytics became so fragmented and what that costs modern businesses. It also illustrates how EDB Postgres® AI (EDB PG AI) solves the one problem the old architecture could not handle: Real-time decisions need live signal, deep history, and transactional truth all at once. EnterpriseDB (EDB) has addressed this by extending open source Postgres into a converged analytics platform that has independent evidence behind it. In a McKnight Consulting Group benchmark, the platform delivered up to 58% lower total cost of ownership than leading cloud data warehouses and held its performance consistently as concurrency climbed.

The central theme is clear: Success will belong to companies that eliminate data movement and consolidate around a shared source of truth for both their analytical engines and AI agents, one that they can fully govern, audit, and control.

History of analytics

How data analytics became fragmented, and why that matters now

Analytics has evolved in waves. Each wave solved a real limitation, and each one added another system to the stack. To understand why "convergence" is now the direction of travel, it helps to walk through how the stack got this fragmented in the first place.

The constant

Every wave solved a problem—and added another system, and another copy of the data to move.

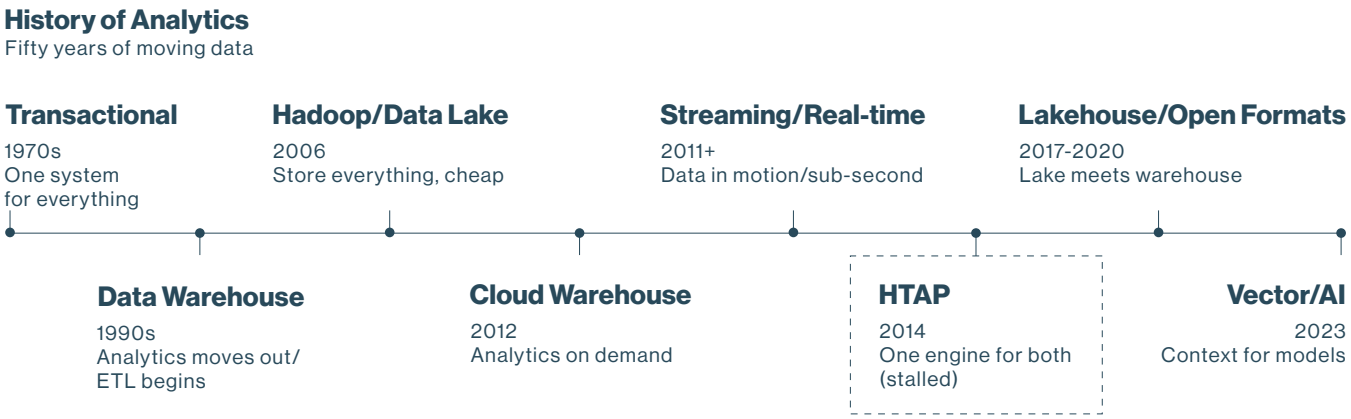


Figure 1. The evolution of the analytics stack, from transactional systems to AI, showing how each wave added another system and marking HTAP as the convergence attempt that did not succeed

In the beginning, one system did everything

For the first couple of decades of business computing, there was no separate analytics tier. Companies ran their reports directly against their transactional databases, with the same systems recording orders, payments, and inventory. Edgar Codd's relational model (1970) and the SQL databases that followed were built for precisely that: fast, reliable reads and writes of individual records. The problem appeared as soon as analytical questions got serious. Scanning millions of rows to summarize a quarter competes for the same resources as the day-to-day transactions the business runs on, and the two workloads begin to interfere with each other.

The data warehouse and the OLTP/OLAP split

The fix, through the late 1980s and 1990s, was to move analytics onto a purpose-built system. Bill Inmon's *Building the Data Warehouse* (1992) and Ralph Kimball's dimensional modeling (1996) defined the discipline, and in 1993 Codd put a name to the divide: OLTP for transactions, OLAP for analysis. Moving data from one to the other required extract-transform-load (ETL), and with ETL came the first data pipelines and a lasting organizational split between the operational and analytical sides of the business. This era also established an assumption that would hold for the next 30 years: Analytics is a separate place you copy data to.

An explosion of specialized systems

The 2000s and 2010s multiplied that pattern. Google's MapReduce paper (2004) and Hadoop (2006) made it cheap to store and process data at web scale, and the data lake was born. Cloud data warehouses—Redshift in 2012, then Snowflake and BigQuery—made large-scale analytics available on demand. Apache Kafka (2011) and stream processors handled data in motion. Column-oriented engines such as Druid and, later, ClickHouse (open sourced in 2016) pushed query latency down to milliseconds for live dashboards. Open table formats, including Apache Hudi and Apache Iceberg (2017) and Delta Lake (2019), and the "lakehouse" label (2020), tried to give the lake the reliability of a warehouse. Most recently, vector databases arrived alongside the large language model wave. Each of these was a genuine advance for a specific problem. Each was also one more system holding one more copy of the data.

HTAP: The first attempt to converge

By the mid-2010s, the cost of all this separation was obvious, and one idea set out to end it directly. Gartner named it HTAP, or hybrid transactional/analytical processing, in 2014. It was a single database that could serve both transactions and analytics, eliminating ETL and letting users analyze operational data in real time. SAP HANA, SingleStore (then MemSQL), Oracle's in-memory option, and TiDB were built around versions of this promise. On paper, it was the cleanest answer imaginable: one engine, one copy, no pipelines.

In practice, HTAP never became the default, and the reasons were structural rather than temporary. Transactions and analytics want opposite things from storage: row layouts and small, fast writes on one side, columnar layouts and large scans on the other. A single engine has to either compromise or maintain both representations internally, which adds cost and complexity. Running heavy analytical queries next to live transactions also invites resource contention, so many HTAP systems ended up routing analytics to a separate in-memory replica—which is, in the end, another copy. And a single engine rarely matched a purpose-built warehouse at petabyte scale or a dedicated columnar engine on real-time ingest. HTAP found real but narrow adoption; the broader market kept its specialized systems.

The lesson outlived the technology. HTAP had the diagnosis right: The split between transactional and analytical data, and the pipelines stitching them back together, is the core problem. Its cure was the issue. Forcing every workload into one engine runs straight into the fact that no single engine is good at everything. That is exactly the line convergence draws. The goal is the same one HTAP chased—to stop copying data between systems—but the method is the opposite. Rather than cram every workload into a single store, use the right engine for each workload over one shared copy of the data.

Best of breed and the swing to decentralization

With the single-engine idea stalled, the industry moved the other way. Teams picked the best tool for each job, e.g., a warehouse for history, a columnar engine for real-time, a document or vector store where it fit. Organizationally, the data mesh concept (Zhamak Dehghani, 2019) pushed in the same direction, though its actual argument was about ownership: Treat data as a product owned by the domain that knows it best, rather than by one central team. Reasonable as that was, combined with best-of-breed tooling it produced environments in which nearly every team and workload ran its own stack.

The sprawl we live in now

The defining feature of today's estate is duplication. Each engine keeps its own copy of the data. Pipelines multiply to move it between them—out of the systems of record into the analytical systems, and increasingly back again, in a round trip the industry named *reverse ETL* around 2020. Data lakes without governance filled up with data nobody could vouch for. Every copy meant another cost, another security boundary, and another opportunity for two systems to disagree about the same fact.

Look across the whole history and one thing stayed constant: At each step, we met a new need by moving data into a new system, not by bringing new capability to the data where it already sat. We centralized into the warehouse and later decentralized into the mesh, but the number of pipelines only grew. For a long time that was an acceptable trade-off. When "analytics" referred to a report someone would read the next day, a few hours of lag and an extra copy or two caused no real harm.

Real-time and AI change that calculation. An automated decision or an AI agent needs current, complete, trustworthy data at the moment it acts, and every copy and pipeline in the way adds latency, staleness, and risk. The cost of moving data, long treated as a minor operational expense, is becoming the factor that decides whether the newest use cases work at all.

That sets up the question the rest of this paper takes on. If no single engine can do everything (which HTAP made clear) but moving data between many engines has become the bottleneck, then neither the monolith nor the disconnected best-of-breed stack is the right answer. The need is specific: a way to run every workload a business has (transactional, historical, real-time, and AI) on one shared, governed copy of the data, using the right engine for each job, without copying that data from system to system. That is what convergence means. The next section looks at why this need, tolerable for decades, has suddenly become urgent.

Problems of a modern business

Why the old trade-off finally breaks

For years, none of this felt like a crisis. Pipelines were a nuisance, not an emergency. Extra copies were a line item. The warehouse ran overnight and the numbers were on someone's desk by morning. You could live with the sprawl.

What changed isn't the architecture but everything around it. Four pressures have been building at once, and together they turn a problem you could ignore into one you can't. They're about money, risk, and whether you can keep up. They are: cost, the hidden tax of moving data; predictability, cloud bills that grow faster than the value they create; control, sovereignty and governance over where data lives and who can touch it; and AI, models and agents that stall without live, governed context. The rest of this section covers each in turn, in that order.

Cost: The tax you never see on an invoice

Ask where a fragmented data estate actually burns money, and the honest answer isn't "the engines." It's the wiring between them. Some of the sharpest engineers in the building spend their days keeping pipelines alive instead of building anything new. And the work is never done, because every schema change and every new source kicks it off again. Meanwhile, the company pays for the same data three times: once to store each copy, once to move and reconcile it, and once more in the salaries of the people babysitting the plumbing. It's a tax that never shows up as a tax. It just quietly eats margin, and it grows with every system you bolt on.

Engine and data sprawl

Every engine keeps its own copy. Pipelines multiply between them. The bill: cost, latency, and drift.

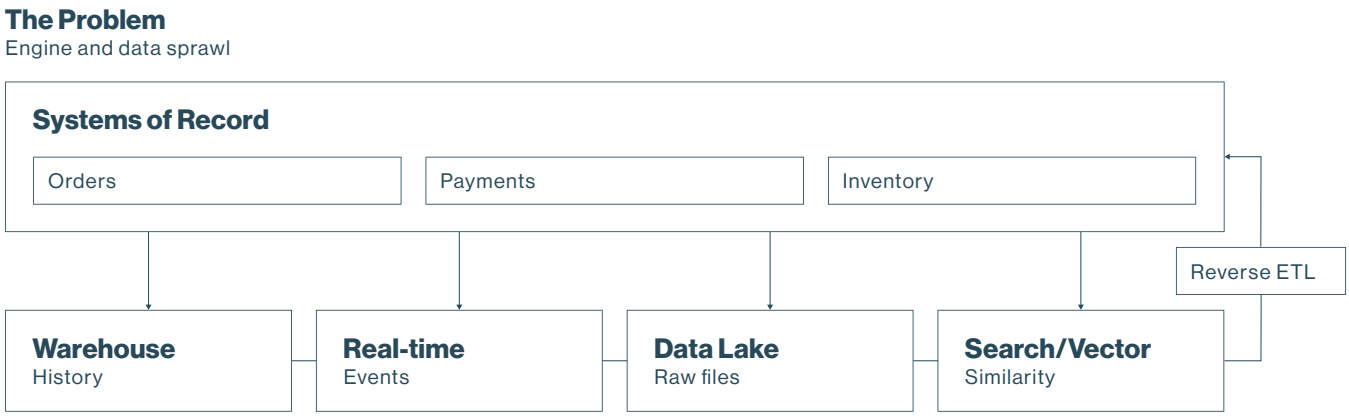


Figure 2. The status quo is systems of record feeding many specialized engines, each keeping its own copy, connected by a growing tangle of ETL and reverse-ETL pipelines.

Predictability: The bill that punishes you for winning

Consumption-based analytics has a cruel logic to it: The more useful it becomes, the more it costs, and the harder it is to predict. Adoption climbs, a few teams ship something great, and next quarter the invoice lands with a number nobody can fully explain. Every new engine is another meter running in the background. Finance can't forecast it, and engineering can't entirely control it. That's why so many companies now quietly admit they're overspending, and why a growing number are hauling workloads back onto infrastructure whose cost they can actually see coming. Analytics costing money was never the problem. The problem is that the cost stopped being something you could plan around.

Control: Who really owns your data

Data has turned political. Where it physically sits, who's allowed near it, and whether you can prove both to a regulator are questions that used to live in a footnote but now land in front of the board and, increasingly, in front of lawyers. Governments are starting to treat control of data and infrastructure as a national strategy, and anyone in a regulated industry can feel the ground shifting. Fragmentation makes all of it harder. Every extra copy is one more place data might sit in the wrong jurisdiction, one more door to lock, one more audit that doesn't quite reconcile. You can't credibly promise to control something that's scattered across a dozen systems and vendors you don't own.

AI: The reckoning

This is the newest pressure, and the one that stings most. AI is only as good as the data it can reach at the instant it acts, which is a far higher bar than any dashboard ever had to clear. A model reasoning from a stale copy, or stalling while a pipeline catches up, doesn't just underperform; it fails in front of a customer. Many AI projects that die do so not because the model was weak. They die on the data layer underneath them.

And the ground is moving fast. The last couple of years have taught the industry that the hard part of building with AI isn't the prompt, it's the context. "Context engineering" has quietly become the real craft: pulling together the right facts, the right history, and the right live signals, and getting them in front of a model at the right moment through retrieval (RAG), feature pipelines, agent memory, semantic layers, and emerging standards such as the Model Context Protocol for wiring agents to tools and data. Strip away the labels and every one of those is a data-access problem in disguise. And every one of them turns slow, stale, and brittle when the context has to be dragged across copies and pipelines before it can be used. Agents don't wait well. The same architecture that merely made yesterday's dashboards a little sluggish makes tomorrow's agents unreliable.

Look past the labels and most of the AI work companies are racing to ship reduces to three problems. One is simplifying a sprawl of stacks into something a team can actually run. Another is searching across everything the organization knows, the mix of keyword and vector search behind retrieval and enterprise search. The third is governing all of it well enough to clear compliance and reach production instead of stalling in review. They look like three separate AI problems. They are the same data-movement problem this paper has been describing, and solving it once, at the data, turns all three into a single job.

Why patching won't work

Here's the trap. These four pressures aren't separate fires you can put out one at a time—they feed each other. The same sprawl that runs up the bill also splinters your governance and starves your AI of context. And they all trace back to a single inherited assumption: the idea, baked in since the very first data warehouse, that analytics is a separate place you ship data off to. Which is exactly why the reflex to add one more engine, one more pipeline, one more contract only digs the hole deeper. This isn't a problem you patch. It's a problem you rebuild around.

That rebuild is what "convergence" really means. It's not a nostalgic return to one giant monolith, and not the disconnected pile of best-of-breed tools everyone's stuck with today, but the right engine for each job, all working over one shared, governed copy of the truth. Fair enough as a principle. But principles don't move anyone. So let's make it concrete with a problem a modern business actually has to solve, and which the old ways simply can't.

The use case you need to solve for

The decision legacy and the cloud both get wrong

For years, the hard real-time decision in payments was fraud: Approve or decline a card swipe in the blink of an eye. Banks and card networks have done exactly that for decades, on architecture that is now decades old. So it's fair to ask what has actually changed. If the old stack could catch fraud in 2005, why can't it carry us forward?

Because the decision itself has changed shape.

The person tapping a card is no longer the only one making payments. Increasingly, the one initiating the payment is an AI agent acting on someone's behalf. It doesn't just get scored after the fact. Instead, it initiates; authorizes; and expects to clear, cross-border, in seconds rather than the two or three days clearing used to take. This increasingly occurs over new rails such as stablecoin settlement. And because a machine is acting on its own, every one of those actions has to carry proof that this agent was authorized to make exactly this payment, for this purpose; and the proof must be checked in the moment and audited forever after. The fraud question hasn't gone away, but it is now the smallest part of a much larger one.

Picture the moment. An agent, acting for a customer, initiates a cross-border payment. You have, give or take, a split second to authorize and clear it, or stop it. To decide correctly, you now have to answer four very different questions at once:

- **Is this consistent with what's happening right now?** The live signal
- **Does it fit the customer and counterparty you've known for years?** The deep history
- **Are the funds and the account actually good this instant, across borders?** The transactional truth
- **Was this agent authorized to make exactly this payment, for this purpose, and can you prove it?** The governed, verifiable record of intent

Fresh + complete + trusted + governed

These are the four characteristics that a fragmented estate can never, in under a second, deliver together.

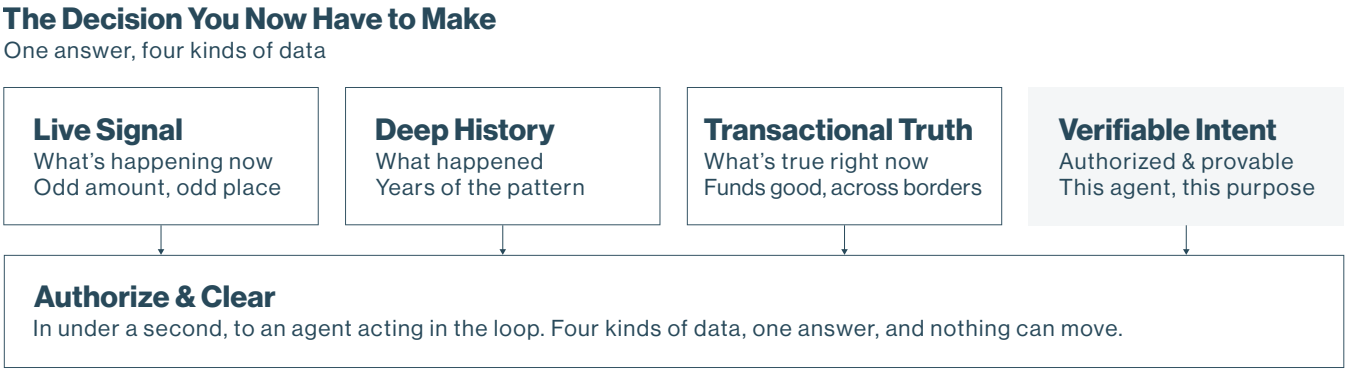


Figure 3. An agent-initiated payment must fuse, in less than a second, the live signal; deep history; transactional truth; and a governed, verifiable record of intent. These are four things a fragmented estate can never deliver together.

Four kinds of data. One answer. Owed in less than a second, to a machine that will act on it immediately.

Why legacy can't do it

The old architecture was built for a world in which clearing took days and a human read the exceptions the next day. The history lives in one system, the live signal in another, the ledger in a third; moving between them takes a batch window. Seconds-level cross-border clearing simply isn't something it can do.

Why the cloud can't either

The cloud data warehouses that replaced it, Snowflake and Databricks among them, are faster, but they solve the wrong half of the problem. The data sitting in them is a copy, in someone else's cloud, often spread across jurisdictions you don't control. They can accelerate a query, but they cannot give you sovereign control over where that data lives and who can touch it, and they cannot enforce that an agent reads only what its authorized purpose permits or produce a provable audit trail at the instant of the decision. You can bolt governance on around the edges. You cannot enforce it where the data actually lives.

Both failures share one root, which is the same one running throughout this paper: Analytics is treated as a separate place you move data to. You cannot clear, govern, or prove in real time what you are still copying from system to system.

Now remember who's deciding

When a person authorized the payment, a slightly stale picture or a loose audit trail was survivable, because someone could catch it later. Hand the decision to an agent acting thousands of times in parallel, with nobody reading over its shoulder, and stale context or ungoverned access stops being a glitch. It becomes a bad payment, or a data breach, at machine speed and scale, with a compliance question no one can answer after the fact. The bar has moved from "a person decides and we reconcile later" to "a machine decides and clears in the loop, and we must be able to prove every decision it made."

This is the use case you need to solve for

It's not because real-time payments are new, but because who makes them, how fast they clear, and what you must be able to prove about them have all changed at once. And you can't fix it by making the pipelines faster, because every copy is another place the data slips out of your control and out of audit. You solve it only when the live signal, the deep history, the transactional truth, and the governance over all of it stop being scattered across systems and become one governed source of truth that every engine, and every agent, can reach at once, without moving anything.

Which leaves an obvious question. If the next decade of decisions has to be fast, complete, trustworthy, and provably governed all at the same time, then who has been building for that, and what does it look like?

So where do we turn?

What we've been building—and the companies who bet on it

We asked who's been building for this. The honest answer, and the reason this paper exists, is that we have.

For more than 20 years, EDB has worked on the least glamorous and most important part of the stack: the transactional core, the system of record that has to be right every single time. We became the largest contributor to Postgres by making the unglamorous parts bulletproof, with the database that runs when everything else depends on it being right. And for the last several years we've been extending that same open, trusted foundation outward, to the analytics and now the AI that sits on top of it.

What came out of that work is convergence made real. Not another engine to bolt on but just the opposite. We built a way for the engines a business already needs—the transactional database, a warehouse for deep history, a real-time engine for live events—to work over *one shared, open source of truth*. Data lands once, in open formats, and every engine reads it in place. No overnight copy. No pipeline quietly shuttling features into the fast store and hoping they're current. The live signal, the deep history, and the transactional truth finally sit in one governed place. We call it converged analytics, and it runs on the most proven open foundation in the industry, rather than the newest one.

We didn't arrive here in a lab. We arrived alongside customers who were already living the problem.

- **A top global card network** was fighting a version of the same real-time gap: The pipeline between its transactional systems and its analytics was slow enough that real-time decisions were always a beat behind the moment that mattered. Collapsing that gap onto shared, live data is the difference between acting on a purchase as it happens and reacting to it the next morning.
- **Kyobo Book Centre**, Korea's largest book retailer, watched its cloud-warehouse bill climb quarter after quarter until the economics simply stopped making sense. Moving onto infrastructure it controlled cut the cost and handed back command of where its data lived and who could touch it.
- **Euronext FX** is a global exchange, so a late answer isn't an inconvenience but a trading and compliance risk. It moved its analytics onto a foundation it owns, across data centers on multiple continents, without missing a step.

The thread running through all of them is the same, and it's worth saying plainly: None of them wanted more tools. In fact, they wanted fewer: one place for the truth, the right engines around it, and one partner accountable for the whole thing. And each got there not by ripping everything out on day one but by starting with the workload that hurt the most and letting the results make the case for the next one.

What we built

Converged analytics, in practice

The idea is easy to say and hard to do: the right engine for each job, all working over one shared, governed copy of the truth. Making it real meant resisting the temptation to build yet another engine and declare victory. Instead, we built EDB PG AI so the engines you already need can stop fighting each other—and we built it on the foundation the industry has trusted longest.

The Shift
From sprawl to a shared source of truth

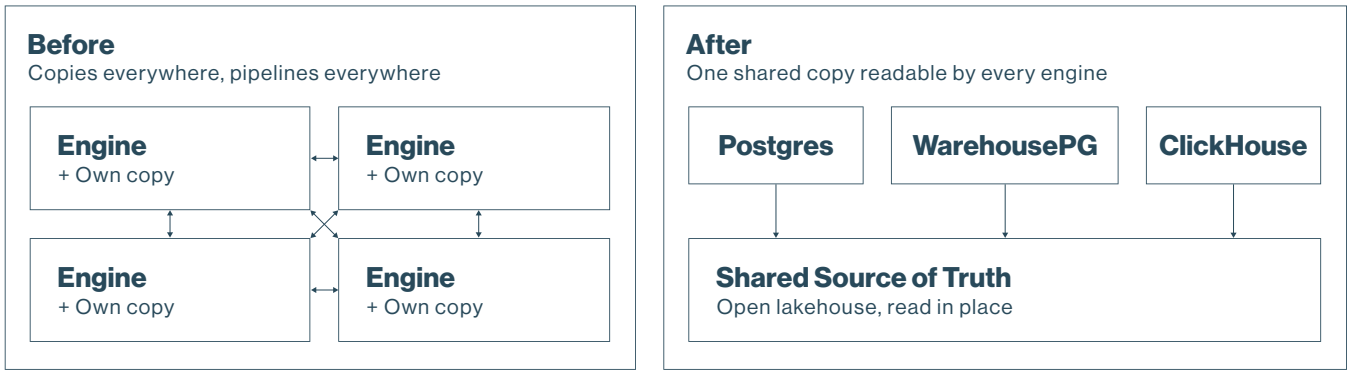


Figure 4. Before and after: many engines with their own copies and tangled pipelines, versus the right engines reading one shared, governed source of truth in place

Start with a foundation that's already proven

Everything sits on Postgres, the transactional core we've spent two decades hardening, and the most widely trusted open source database in the world. This strategy represents the entire philosophy in brief: The solution for an AI-powered, real-time future is not to find a flashy new engine but to take the world's most reliable open foundation and expand its capabilities. Postgres remains the system of record, the place that always knows what's true right now.

One shared source of truth, in open formats

Around that core, we put an open lakehouse: Data landed once, in open table formats (Apache Iceberg), on ordinary object storage, catalogued in one place. This is the single most important move in the whole design. Instead of each engine hoarding its own copy, there is one governed copy that every engine reads *in place*. The catalog—open, not proprietary—is where governance lives, so access, lineage, and audit are defined once and travel with the data rather than being re-implemented in every tool. No overnight export. No pipeline shuttling features between systems and praying they're fresh.

The right engine for each question, over the same data. With a shared source of truth established, the engines become specialists that cooperate instead of silos that compete.

- **Postgres** answers "What's true now?": the live operational record.
- **WarehousePG**, a Postgres-native massively parallel warehouse, answers "What happened?": deep history, complex joins, in-database ML at petabyte scale.
- **ClickHouse**, the newest member of the family, answers "What's happening right now?": sub-second analytics over billions of live events.

Crucially, all of them work over the same Iceberg tables rather than keeping their own copies. Postgres and WarehousePG read and write those tables; ClickHouse reads them natively today, with write support on the roadmap. Postgres and ClickHouse can even read each other directly. The data doesn't move to the engine; the right engine is simply pointed at the data.

PGAA: The part that makes it converge

The piece that unlocks true convergence on EDB PG AI is Postgres Analytics Accelerator (PGAA). Its principle is simple: Postgres stays in front, any engine can work behind it, and the SQL never changes. An analyst or an application issues ordinary Postgres SQL; PGAA decides how and where to satisfy it against the open lakehouse, accelerating queries and, when needed, pushing the heaviest work out to scale-out and GPU-backed execution. The point isn't the plumbing (that lives in the appendix, for those who want it). The point is what it removes: the copies, the pipelines, and the lag between them.

The right engine for each job

Over one shared, governed copy of the data.

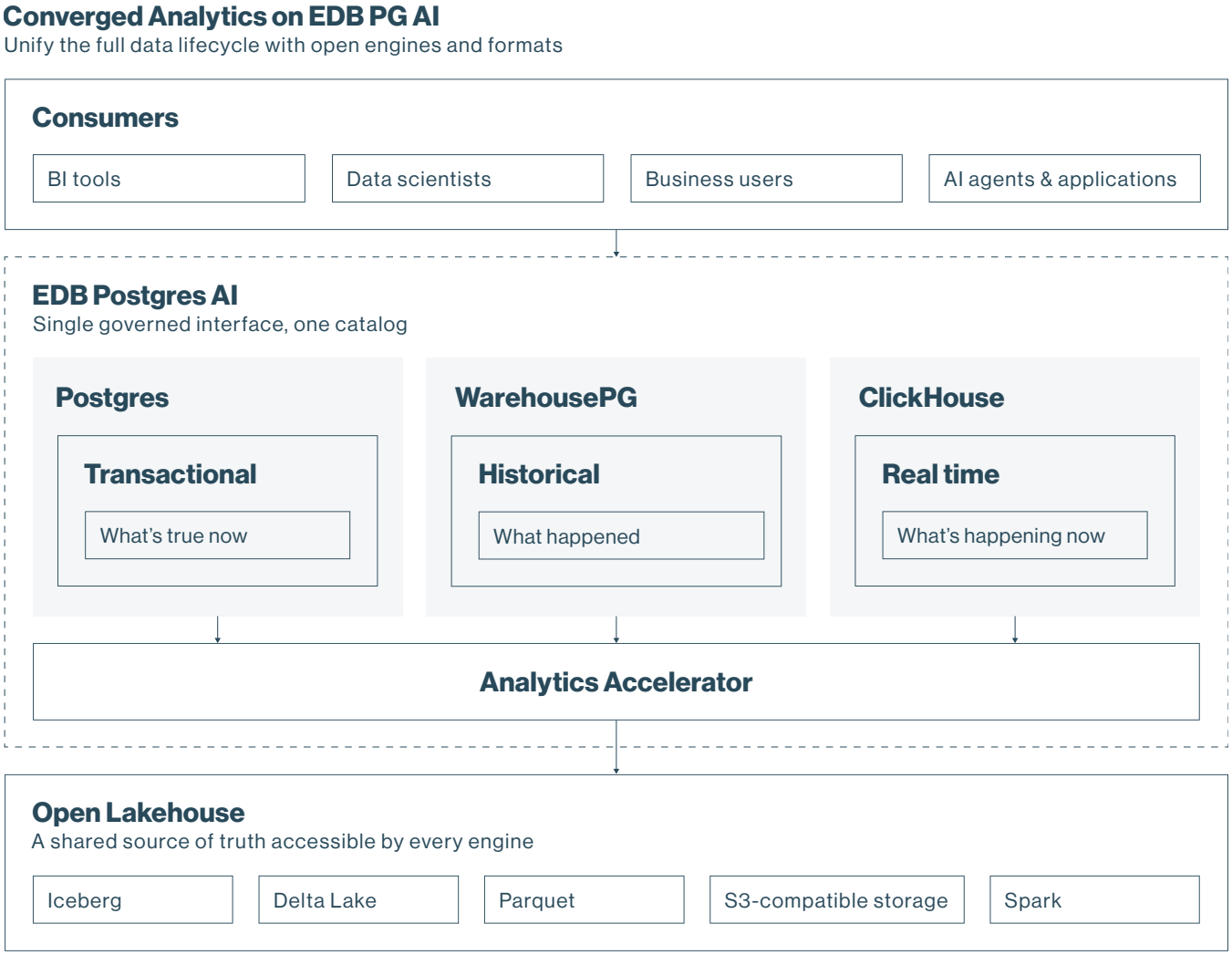


Figure 5. The converged analytics platform: Consumers over EDB Postgres AI, Postgres, WarehousePG, and ClickHouse above the PGAA accelerator, all reading an open lakehouse foundation.

Governed once, owned everywhere

Because the whole thing is built on open source and open formats, it runs where the business needs it to—on-premises, in a sovereign data center, or in any cloud—without changing the data, the SQL, or the governance model. You are not renting a proprietary engine you can never leave. You own your strategy and your infrastructure. That is what makes the sovereignty and cost pressures from earlier in this paper answerable rather than merely acknowledged.

Governance enforced at the data, not bolted on around it

This is where the fourth requirement from the payment decision, the proof that an agent was authorized for exactly this purpose, actually gets satisfied. In most stacks, governance lives in a layer the application is trusted to call; an agent that reaches the data by another path simply isn't checked. Because everything here reads one governed copy through one catalog, that check moves to the only place it can't be bypassed: query execution itself. An agent's authorized purpose travels with its request, and the engine returns only the data that purpose permits, no matter what the agent asks for. Every access is written to a session-level provenance log that records what was read, under which session, and which authorization allowed it, so "prove it" has a real answer months later, instead of a shrug. The data stays encrypted at rest and under the customer's control. Taken together, that is the governance control plane regulated buyers increasingly ask for: converged analytics and AI, governed and fully auditable, enforced where the data lives rather than around its edges.

Now look back at that payment decision

The call the old architecture couldn't make—live signal, deep history, transactional truth, and provable authority, fused in under a second, for an agent acting on its own—is exactly what this design is built for. The live signal is already in ClickHouse. The years of history are already in WarehousePG. The funds and current account standing are already in Postgres. And the agent's authorized purpose is enforced and logged against the same governed data every engine reads. There is no pipeline to wait on, no stale copy to be wrong about, and no governance gap for an autonomous agent to slip through. The question, the answer, and the proof finally live in the same place, both for a human analyst today and for an AI agent acting in the loop tomorrow.

That's EDB PG AI. The natural next question is whether it actually pays off when a business commits to it. For that, we'll let the numbers, measured by someone independent, speak.

What the numbers say

An independent benchmark, and why these metrics matter for converged analytics

We'll be the first to admit that a vendor quoting its own performance numbers is worth about what you'd expect. So we didn't quote our own. We asked McKnight Consulting Group to run EDB PG AI through a standard, deliberately punishing benchmark of 10 terabytes of TPC-DS at high concurrency against the cloud data warehouses most teams are actually choosing between: Snowflake, Databricks, Redshift, and Hive-on-Iceberg as an open source baseline. Here's what they found. And, more to the point of this paper, here's why these particular numbers are the ones that matter for the converged, real-time future we've been describing:

- **Up to 58% lower total cost of ownership** than leading cloud data warehouses
- **$222,886 a year versus $351,953** for a comparable Snowflake multi-cluster configuration, about $129,000 saved on a single 10 TB workload, and those savings repeat on every workload you move
- **The most consistent performance under load** of everything tested: a 2.7x slowdown at 5x concurrency, where the cloud warehouses degraded 3.9–4.1x

Cost you can actually plan around

Think back to the bill that punishes you for winning. The value of 58%, and of a number such as $222,886, isn't only that it's smaller—it's that it's *knowable in advance*. Capacity-based pricing means the figure doesn't lurch every time adoption climbs or a new team comes online. For a CFO, that quietly reclassifies analytics from an unpredictable variable into a line you can budget. The saving is nice; the predictability is the point. And $129,000 is a single 10 TB workload. A large enterprise runs dozens of them, so that 58% compounds into one of the biggest line items on the data budget, not a rounding error on one project.

Performance that holds when everyone shows up at once

This is the metric that matters most for convergence, and it's the easy one to skim past. Convergence means everything leans on the same source of truth at the same time, including BI users; data scientists; and, more and more, fleets of AI agents querying in parallel around the clock. High concurrency isn't an edge case in that world; it's the normal operating condition. A platform that slows 2.7x under 5x load while others slow closer to 4x is the difference between a shared foundation that scales into that future and one that buckles the moment the agents arrive.

And the honest part

McKnight didn't hand us a clean sweep, and the caveat is worth quoting because it actually sharpens the case. The report notes that cloud warehouses "suit high-performance analytics for the most demanding queries," while EDB PG AI "works efficiently for the high-concurrency analytics that power daily operations ... revealing the merits of a hybrid approach." Fair. For a single, exotic, once-a-day monster query, a specialized cloud warehouse may still edge ahead. But that isn't what runs up your bill, and it isn't what an agent generates a thousand times an hour. What dominates real operations, and the many-actors, real-time world we're heading into, is high-concurrency, everyday analytics. That is precisely where predictable cost and predictable performance line up on the same side.

Strip it all the way down and the benchmark is really measuring two kinds of predictability: what you'll pay, and how it will hold up when the whole business, and its agents, lean on it at once. Those happen to be the exact two guarantees a shared source of truth has to make. Which brings us back to where we started, and to where all of this is quietly heading.

Where this goes next

Step back from the last 50 years and the history of analytics looks less like a straight line and more like a pendulum. Into the warehouse, out to the lake. Centralize under one team, decentralize into domains. One engine meant to do everything, then a separate engine for every job. Each swing corrected the last one's excess, and each swing left behind more copies of the data and more pipelines to move it. The one thing that never changed was the assumption underneath all of it: that data has to keep moving from one system to the next in order to be useful.

That assumption is now the thing holding businesses back. For decades the cost of moving data was tolerable, because the deadlines were forgiving: a report by morning, a dashboard by Monday. The work arriving next isn't as generous. Decisions are moving into software, and increasingly into agents that act on their own, in the moment, thousands of times an hour. They need current, complete, trustworthy context at the instant they act. Every copy and every pipeline between the question and the answer has stopped being an inconvenience and become a liability.

So the next move isn't another swing of the pendulum. It's to stop the pendulum, to stop shuttling data between systems and let the right engine for each job work over one shared, governed copy of the truth. That is not a bet on the newest technology. If anything, it's the opposite. The businesses that get this right will be the ones that build on the most proven open foundations rather than the most fashionable ones, because the systems that carry a company through the next decade tend to be the ones that already proved they could carry it through the last.

Which leaves a question worth sitting with. The systems your business depends on most—the ones that can't be wrong, can't go down, and can't leak—already run on foundations proven over decades, not months. As analytics and AI become just as mission-critical, why would you hold them to a lower standard? The companies that answer that well won't get there by adding one more system to the pile. They'll get there by converging on one. If that's the direction you're ready to move toward, the foundation already exists, and it's worth finding out who built it.

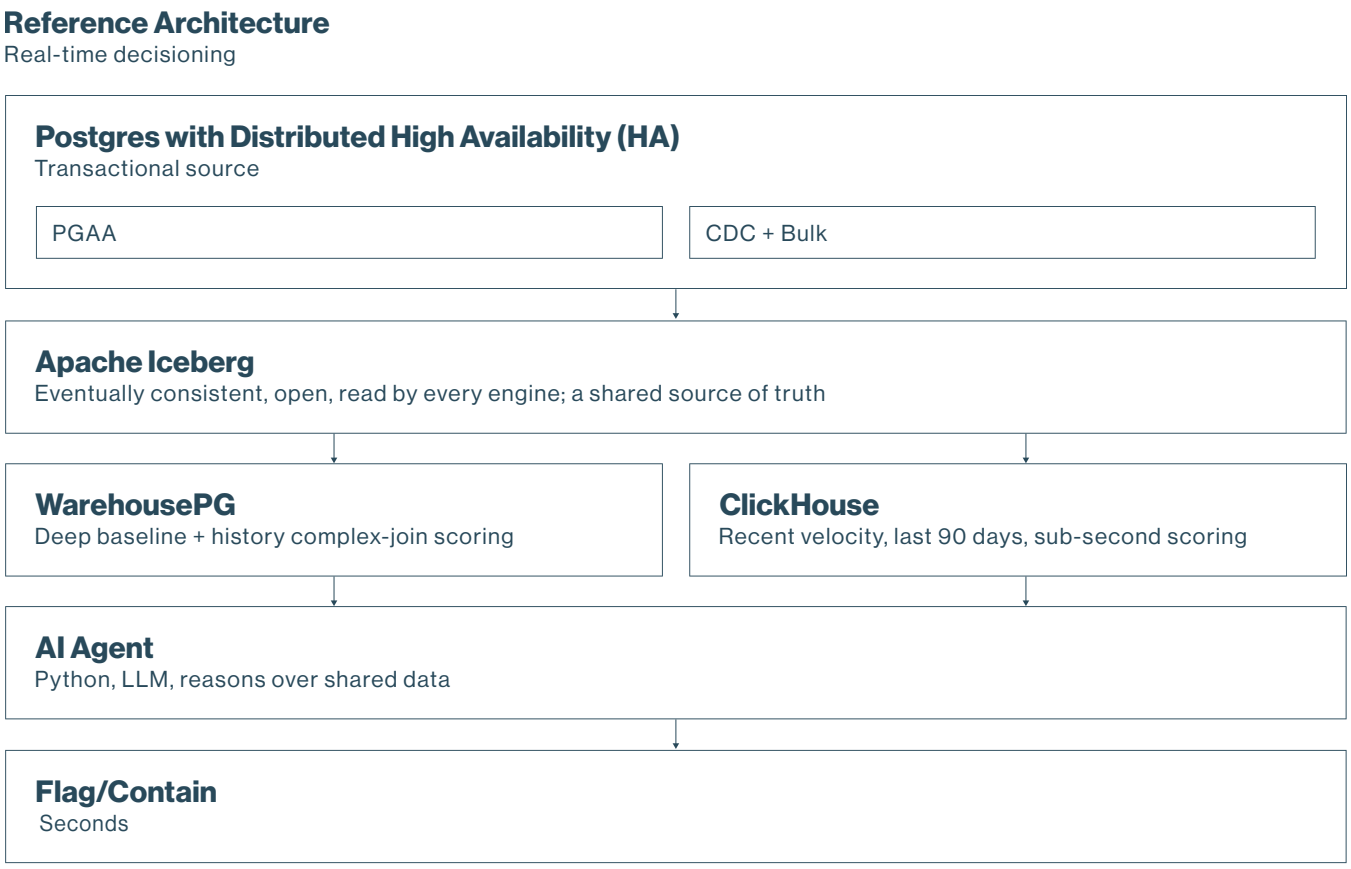
To learn more about EDB PG AI, please visit www.enterprisedb.com/products/converged-analytics.

Appendix: How it works

The body of this paper keeps the mechanics light on purpose. This appendix is for the reader who wants to see how converged analytics actually fits together enough to evaluate it, or to reproduce the reference design behind the real-time example used earlier. It is deliberately concise; it is not a full product manual.

A.1. A reference architecture

The real-time decision described earlier—score a live event against deep history and current transactional state, in seconds—maps to a concrete, deployable design. The version below runs end to end on Docker Compose.



- **Postgres with Distributed HA** is the transactional source of record. Powered by EDB Postgres Distributed (PGD), its changes stream into the lakehouse continuously (via change data capture), with bulk loads where a full copy is needed.
- **Apache Iceberg**, on object storage, is the single shared source of truth—one governed copy, in open format, read by every engine.
- **WarehousePG** holds the deep baseline and historical context and handles the complex-join scoring.
- **ClickHouse** scores recent, high-volume activity sub-second.
- An application or **AI agent** issues standard SQL (or calls the engines directly) and returns a decision, reasoning over the same governed data rather than a private copy.

A.2. The components

PGAA (Postgres Analytics Accelerator): PGAA presents a standard Postgres SQL surface and routes execution to the right place against the open lakehouse. The contract for the caller never changes; what changes is how the query is satisfied.

Mode	Mechanism	When it's used
Real-time replication	Change stream from Postgres into Iceberg (via PGD)	Keep the lakehouse current
Bulk offload	`CREATE TABLE AS` into open formats	Move large history on demand
Single-node acceleration	Apache DataFusion, in-process	Vectorized speed-up on one node
Scale-out / GPU	Apache Spark via Spark Connect	Heaviest jobs, distributed and GPU-accelerated

WarehousePG: A Postgres-native, shared-nothing MPP warehouse. A coordinator plans queries (with a standby for high availability) and segment hosts execute them in parallel. PGAA runs inside every segment, each with its own DataFusion instance, so scans against Iceberg happen at full parallelism. It is transactionally consistent and built for depth and accuracy at petabyte scale. Because it is Greenplum-compatible, existing SQL and skills transfer directly.

ClickHouse: A distributed, column-oriented OLAP engine, meaning one logical distributed table across shards and replicas, coordinated by ClickHouse Keeper, on MergeTree storage. It ingests high-volume event streams and serves sub-second queries to many concurrent users. It connects to the Postgres family through an `ENGINE = PostgreSQL` table engine, reads the shared Iceberg tables natively today, and is eventually consistent by design (write-to-Iceberg support is on the roadmap).

The shared lakehouse: Data lands once in open table formats (primarily Apache Iceberg) on S3-API object storage (for example, MinIO). It is catalogued through an open Iceberg REST catalog (such as Lakekeeper). The catalog is where governance lives. Access, lineage, and audit are defined once and apply to every engine that reads the data.

A.3. Consistency model

The design is explicit about consistency rather than blurring it. The lakehouse is the shared, *eventually consistent* source of truth. Real-time work runs in ClickHouse (eventually consistent, sub-second). Accuracy-critical work, such as large joins and transactions, runs in WarehousePG (transactionally consistent) or against Postgres itself. Choosing the engine for a workload is therefore also a decision about the consistency it needs.

A.4. Deployment

Because every layer is open source over open formats, the same architecture runs on-premises, in a sovereign data center, in public cloud, or on GPU-accelerated hardware. It does not change the data, the SQL, or the governance model. That portability is what makes the sovereignty and cost arguments in the body of this paper practical rather than theoretical.

EDB POSTGRES AI

EDB Postgres AI is the first open, enterprise-grade sovereign data and AI platform, with a secure, compliant, and fully scalable environment, on premises and across clouds. Supported by a global partner network, EDB Postgres AI unifies transactional, analytical, and AI workloads, enabling organizations to operationalize their data and LLMs where, when, and how they need them. For more information, visit enterprisedb.com