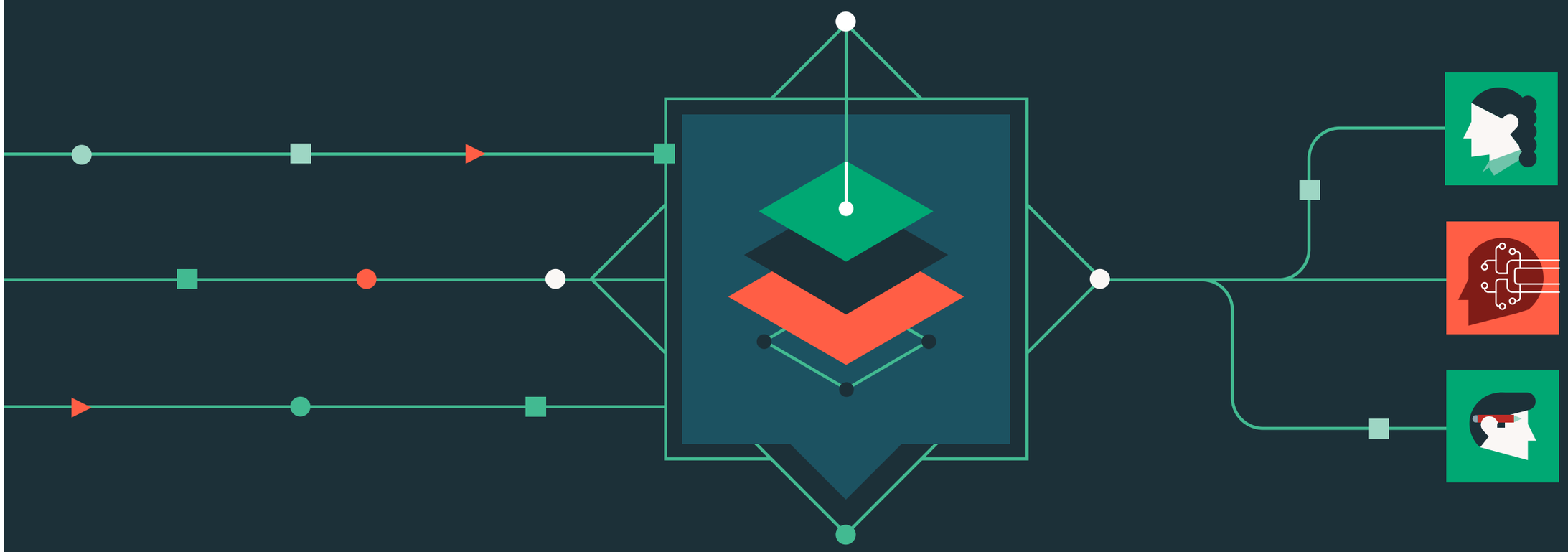
databricks

eBook

Redefining the Modern Semantic Layer

Shifting business metrics into your data platform
to enable unified insights for BI and AI



Contents

Introduction

The cracks in today's data foundations are visible everywhere. Revenue numbers shift depending on the tool used to calculate them. Meetings open with reconciliation instead of decision. AI assistants answer instantly but with definitions that do not match finance reports.

These are not minor annoyances. They represent decision debt that compounds until leaders lose trust in the numbers and hesitate at the very moment they need to act with confidence.

For years the industry has turned to semantic layers as the solution. The idea of a shared language between business and data is widely accepted. Yet the versions of semantic layers most companies know were designed for a different era. They lived inside business intelligence (BI) tools, tied to proprietary languages, and struggled with scale, sprawl and the new demands of AI.

This book is about a new approach. By shifting semantics into the data platform itself — next to the data, policies and lineage — we create a foundation that is both governed and reusable. From there, definitions can project outward to every dashboard, notebook and AI agent. This is not another book about semantic layers as they have always been. It is a guide to platform-native semantics that deliver consistency, speed and trust across the modern data stack.

In this eBook, you'll learn:

- Why traditional semantic layers break down under modern scale, fragmented tools and the rise of AI
- How shifting semantics into the data platform creates consistency, speed and governance by default
- The five principles that guide successful semantic layer design and prevent common pitfalls
- How Unity Catalog delivers platform-native semantics through metric views, lineage, policy and performance services
- How AI agents and knowledge stores build on semantics to deliver trustworthy answers that continuously improve
- How to put semantics into practice, with examples, links to documents, demos and training

What Is a Semantic Layer (and Why Does It Matter)?

Without a semantic layer, most organizations live with quiet chaos. Definitions are scattered across dashboards, hidden SQL queries, spreadsheets and notebook cells. Two teams ask for “ARR” and get two numbers. A weekly review opens with “What did you pull?” instead of “What did we learn?” An AI assistant answers confidently but contradicts the finance report because it queried a different model. The result is decision debt: ambiguity that compounds into rework, delays and missed opportunities.

You can see the pattern everywhere. In a SaaS business, annual recurring revenue (ARR) run rate sounds obvious until people discover different trailing windows in different tools. In e-commerce, net revenue swings based on how returns, taxes and discounts are treated. Analysts spend more time reconciling than exploring. Leaders lose trust in the numbers and slow their decisions.

A semantic layer addresses this by creating a single, shared language for the business and the data. It names and precisely defines the things leaders care about, then makes those definitions reusable everywhere: in BI dashboards, in SQL editors and data science notebooks, and increasingly in AI agents that answer questions in plain English.

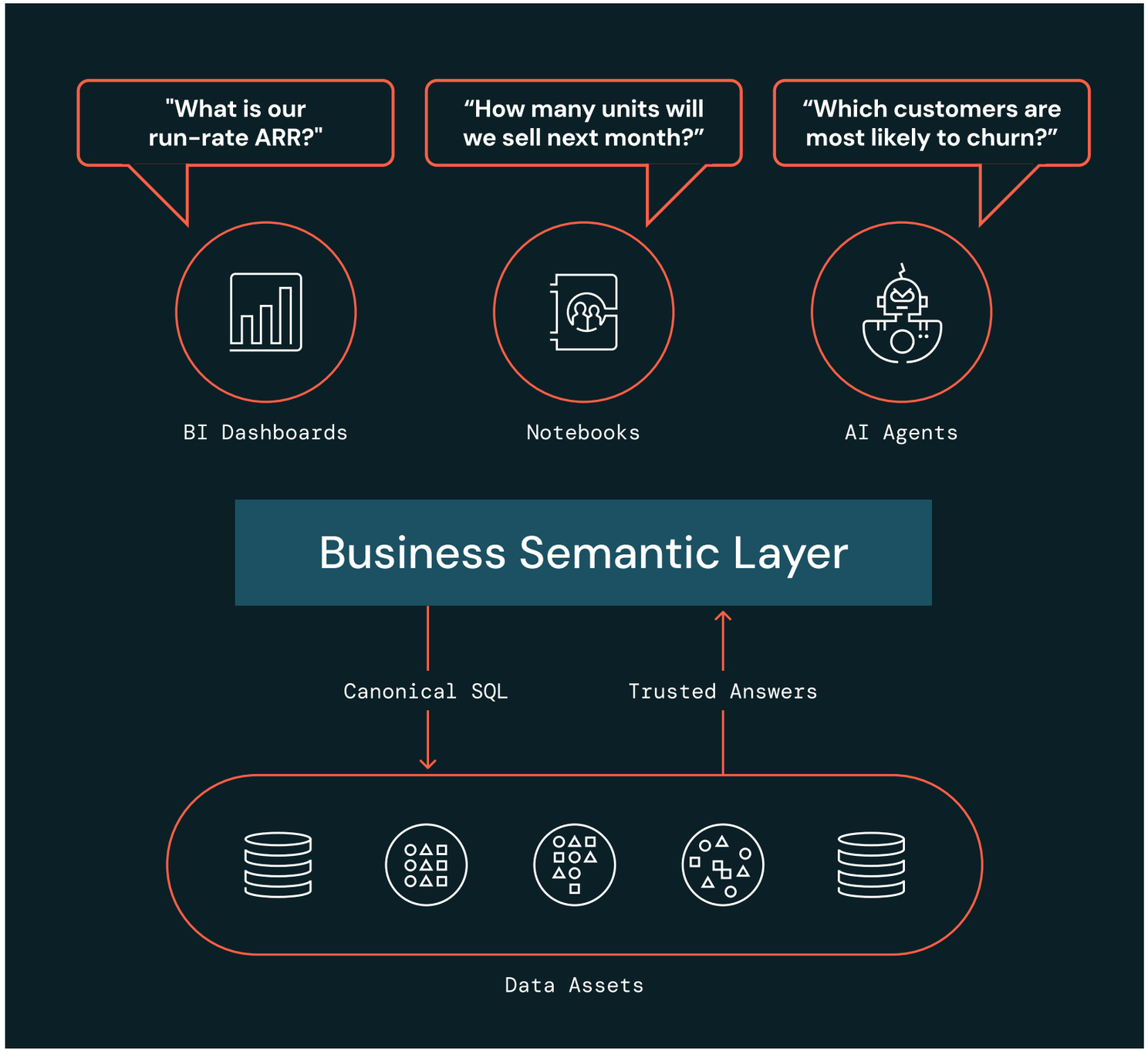


Figure 1. A semantic layer turns business questions into gold standard or canonical SQL queries that deliver consistent and trusted answers from your data.

Good semantics are pragmatic, not precious. They define just enough to answer core questions repeatedly and correctly, then evolve as the business changes. They should not feel like another tool to learn. They should feel like an institutional habit made concrete in the same platform where data and governance already live.

In practice, this produces three outcomes:

1. **Shared language:** Metrics, dimensions and relationships have names and owners that everyone recognizes
2. **Reuse by default:** The same definitions appear in dashboards, ad hoc SQL, notebooks and AI agents, so answers line up
3. **Safer speed:** Changes propagate in a controlled way, so teams move faster without breaking trust

Signals you need a semantic layer

- Review meetings start with number reconciliation
- The same metric has multiple “official” definitions
- Analysts copy logic between tools and fix bugs in more than one place
- AI answers vary by interface because each interface hits a different model
- New questions require model surgery instead of simple reuse

When these signals fade, conversations improve. Stakeholders stop debating definitions and start asking better questions. Analysts shift from translation to discovery. AI becomes more helpful because it can speak the business’s vocabulary with precision. That is the promise of a modern semantic layer and the foundation for the rest of this book

What the semantic layer actually contains

At a minimum, a usable semantic layer comprises:

- Business-friendly names and descriptions for key entities and events (e.g., customers, orders, subscriptions)
- Metrics and their measures (e.g., NRR = growth minus churn; Churn Rate = churn_amount / starting_arr)
- Dimensions and hierarchies (e.g., Region → Country → State; Product Category → SKU; Calendar → Fiscal Month → Quarter)
- Time intelligence (e.g., trailing windows, cumulative totals, period-over-period)
- Lineage, ownership, certification and policy (RLS/CLS) that travel with the definition
- Interfaces for reuse: SQL, notebooks, dashboards and — increasingly — AI agents

Add to that the domain language — synonyms, acronyms, display rules — and you have something both humans and machines can use without guesswork.

A semantic layer turns raw data into a shared language — the precondition for consistent decisions and reliable AI.

A Short (Honest) History of Semantic Layers

The semantic layer is not a static concept. It has evolved with BI itself. Each era has reshaped how organizations build a shared understanding of their data, driven by the technical realities and analytical ambitions of the time. This history can be seen as a series of waves, each attempting to solve the fundamental challenge of translating raw data into business meaning.

The Semantic Graph and the Universe (1990s)

The first commercial semantic layers emerged in the early 1990s. MicroStrategy built its platform on the powerful "Semantic Graph," while BusinessObjects introduced the revolutionary "Universe." Both aimed to solve the same problem: mapping complex relational tables and joins into business-friendly terms. For the first time, a non-technical user could ask a question without knowing SQL. This was a breakthrough, successfully separating the “what” of a business question from the “how” of a database query.

The OLAP cube era (late 1990s–2000s)

As data volumes grew, querying relational databases directly became too slow for interactive analysis. The industry’s solution was the multidimensional online analytical processing (OLAP) cube. Engines like Oracle Essbase and Microsoft Analysis Services pre-aggregated data into rigid but high-performance structures. These cubes became the era's de facto semantic models, embedding hierarchies, measures and calculations directly into the database. Languages like MultiDimensional eXpressions (MDX), and later Data Analysis Expressions (DAX), provided a formal way to query these structures, making the cube the center of the analytical world.

The enterprise model (2000s–2010s)

With the rise of large-scale enterprise BI, the focus shifted to governance and consistency. This was the era of the IT-managed, centralized model. Tools like Cognos Framework Manager allowed teams to build a single, sanctioned version of the truth. These layered models separated physical data connections from business logic and presentation, ensuring everyone used the same definitions. This approach delivered consistency, but it was often rigid and slow to adapt to new business requests.

Semantics as code (2012–2019)

The arrival of powerful cloud data warehouses like Amazon Redshift and Google BigQuery changed everything. The need for pre-aggregated cubes diminished. Looker, founded in 2012, pioneered this shift with LookML, which treated the semantic model as code. Definitions were written in a simple modeling language, version controlled with Git and used to generate live SQL queries against the warehouse. This empowered data analysts, moving model creation away from central IT and toward teams closer to the business.

The universal layer (2019–present)

As organizations adopted a diverse ecosystem of BI tools, the old problem of fragmentation re-emerged in a new form. The response was the universal semantic layer. Decoupled from any single BI tool, these "headless" platforms define logic once and serve it to many clients via APIs (such as SQL, JDBC and REST). Tools like Cube, AtScale and dbt's Semantic Layer embody this vision of a shared semantic hub for the entire data stack, from Power BI to a Python notebook. This wave further refines the idea into a metrics layer, focusing narrowly on standardizing the most critical definitions — key business metrics — in a portable, declarative format.

What the waves taught us

Each wave has tried to resolve the same essential tension: balancing the need for governance with the demand for agility. The story of the semantic layer is a pendulum swinging between centralized control and decentralized access, constantly adapting to the underlying technology. From universes to cubes, code to APIs, the goal has remained the same: to create a shared language for data. And as each wave leaves unfinished business, the next is already forming, promising a better, more flexible way to bridge the gap between data and decision.

Why Yesterday's Approaches Crack Under Today's Workloads

Every wave of semantic layers solved the problems right in front of it. Universes made SQL-heavy schemas intelligible. Cubes delivered governed calculations at scale. IT-managed models promised stability. Code-first approaches gave analysts speed. Headless and metrics layers tried to knit things back together across tools.

But the reality is that each of these approaches starts to bend and eventually break when confronted with the workloads we face today.

Scale and speed no one planned for

Early universes and cubes were designed for a world where data was smaller, change was slower and BI meant reports that refreshed overnight. Today's workloads are different. A single query might scan billions of rows, and users expect answers in seconds. Aggregate tables and cube processing schedules — once clever engineering — often become brittle workarounds when data is always changing and freshness matters as much as speed.

Too many versions of the truth

For every generation, the same problem reappears: the truth lives in too many places. There is a universe definition here, a cube calculation there, LookML in one tool and DAX in another. Each may be consistent on its own, but across tools, the cracks show quickly. Two teams ask the same question and get different numbers, not because the math is wrong but because the definitions diverged. What began as an elegant way to simplify one surface becomes fragmented when an enterprise has many surfaces.

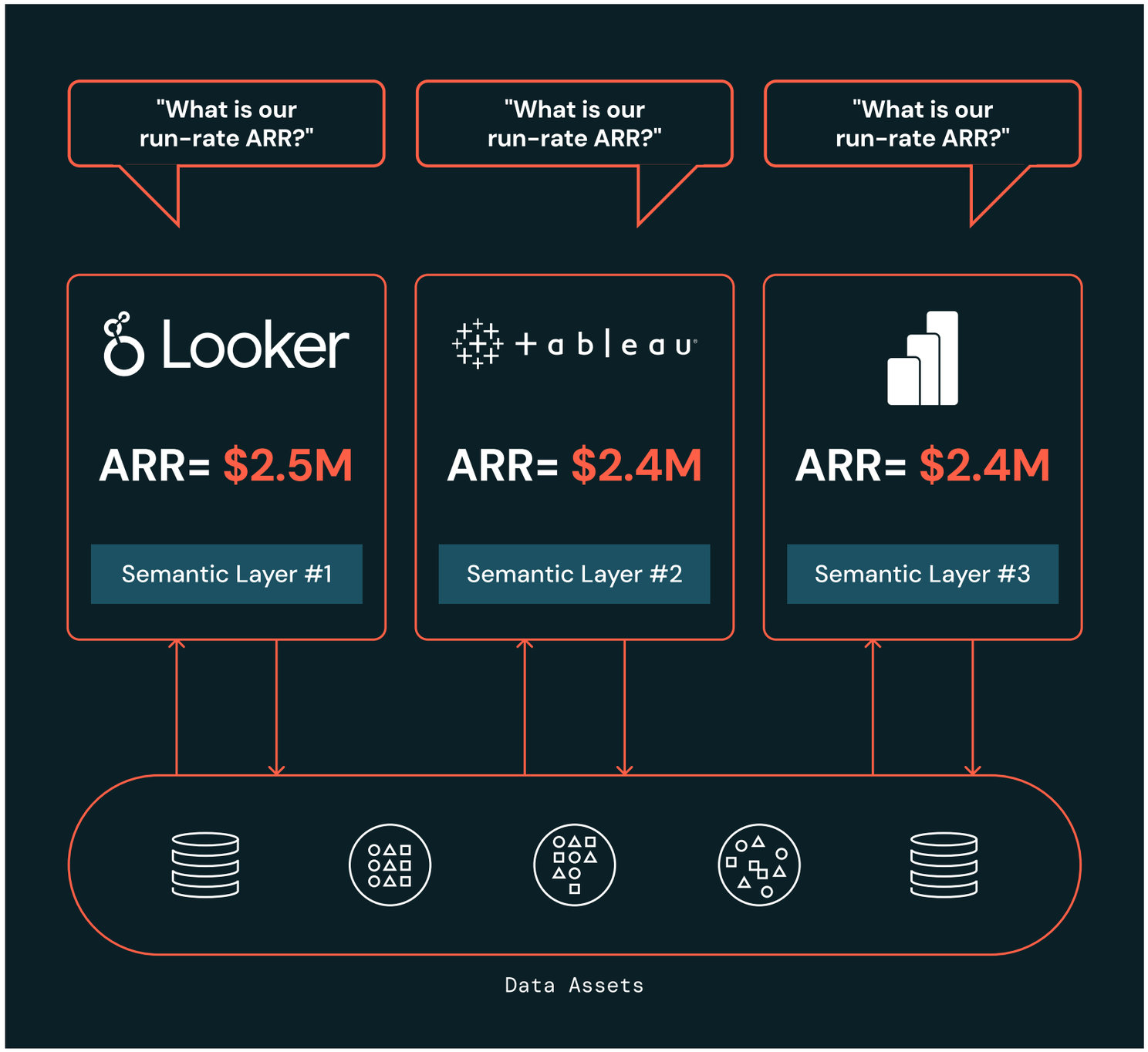


Figure 2. Each tool has its own semantic layer, leading to inconsistent and conflicting answers.

The biggest risk isn't a wrong number. It's a system where the right number depends on where you ask the question.

Semantics trapped in tool silos

The problem is reinforced by the fact that every BI tool has its own proprietary language — DAX in Power BI, LookML in Looker, VizQL in Tableau, MDX in the cube era. These languages were powerful innovations, but they tied semantics to a single surface. What should have been universal business logic instead became locked inside the walls of one tool. That lock-in not only fragments dashboards, it blocks reuse entirely. Data scientists in notebooks, operational apps and AI agents are all left outside, unable to access or extend those definitions.

Governance in two worlds

Security, lineage and policies belong with the data itself, but when semantics live outside the platform, governance has to be reinvented in every tool. Row-level security might be set one way in a dashboard, another in a cube and not enforced at all in a metrics API. Auditors, analysts and now AI agents all face the same question: which set of rules actually applies? That duplication is not just inefficient — it's dangerous.

Performance patched, not shared

Performance, too, gets handled piecemeal. Every tool builds its own caches, extracts or acceleration strategies. They work locally but multiply the burden globally. Keeping dozens of disconnected caches in sync with the warehouse becomes its own operations problem — and one more place where numbers don't quite match.

Not built for AI

Perhaps the most important gap of all is that none of these semantic layers were originally designed with AI in mind. Large language models (LLMs) don't just need a metric definition, they need context. They need synonyms, explanations and guardrails to avoid hallucinations and build trust. A dashboard can tolerate some ambiguity, but an AI agent amplifies it into misinformation. When semantics live outside the platform, disconnected from governance and lineage, AI agents inherit those cracks. Instead of answering with confidence, they risk giving answers that are plausible but wrong — and hard to audit with confidence.

The repeating story

If you zoom out, you notice a pattern. Each generation solved the problems it could see — simplifying SQL, governing calculations, speeding up queries — but always from the outside in. Semantics were bound to the tool, not the platform. That made sense when BI meant a single reporting system, but in a world of cloud platforms, multi-tool ecosystems and AI-driven consumption, it's no longer enough.

The lesson is clear: semantics can't sit on the edges anymore. When they do, they duplicate governance, fracture definitions, struggle with scale — and leave us unprepared for AI.

The Answer: Shift Semantics Into the Data Platform

The durable fix: manage business semantics **inside the data platform** — next to data, policies, lineage and audit — and project them outward to every tool and agent. Think of semantics as shared infrastructure: governed assets that multiple teams and systems can rely on. This isn't just architectural preference. It's a **unified semantic layer** that lays the foundation for reliable AI and sustainable analytics at scale.

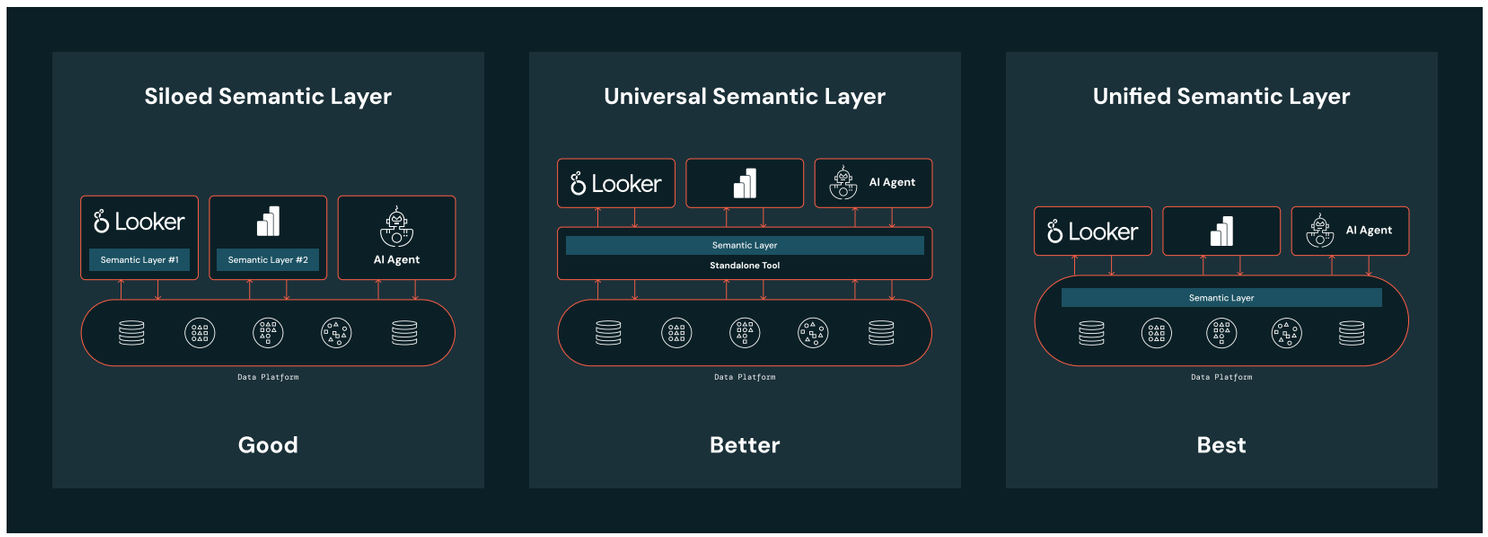


Figure 3. A unified semantic layer ensures every tool and AI agent works from the same trusted definitions, governed all in one place.

5 key principles that will keep you out of trouble

These principles distill the lessons learned from hundreds of semantic layer implementations across different industries and scales:

1. **Author once, reuse everywhere:** Treat definitions as **platform-native assets**, not payloads embedded in charts or dashboards. A metric like customer lifetime value lives in one place with one owner, then serves any dashboard, notebook or agent. This removes the translation tax and makes consistency the default.
2. **Proximity to governance:** Access controls, masking, lineage and certification **travel with the asset**. If a metric inherits **row- and column-level** policies from its sources, those policies apply everywhere — dashboards, AI agents and APIs. Governance becomes infrastructure, not documentation.
3. **Open by design:** Prefer SQL and published APIs (REST, GraphQL, JDBC, ODBC) and **avoid proprietary domain-specific language (DSL) lock-in**. The semantic layer must be consumable by today's and tomorrow's tools, including non-BI surfaces.
4. **One source for humans *and* AI:** Dashboards and conversational agents read from the same source. AI does need extra context — synonyms, explanation templates, guardrails — so attach that **as metadata** to the asset. "What's our NRR this quarter?" returns the same answer whether plotted or spoken.
5. **Semantics as code:** Treat definitions like code — **versioned, reviewed, promoted** through dev → staging → prod. Changes are testable, auditable and deployed via the same continuous integration and continuous delivery (CI/CD) rigor that governs other critical infrastructure.

Security, policy and audit by construction

When semantics live in the platform, **agents don't need their own security layers** — they inherit platform policies. The same row-level security that restricts a sales dashboard to its territories applies when an AI agent answers questions about those metrics. Lineage answers "where did this number come from?" and "who changed what, when?" in real time, not at quarter-end. Certification becomes a **first-class signal** trusted by humans and AI.

Performance as a shared service

Acceleration — materializations, aggregates, caching — belongs **once** in the platform, where every consumer benefits. Instead of each tool inventing its own cache, the platform maintains multiple levels of pre-computed aggregates and **smart-routes** queries to the most efficient source. A business analyst exploring "monthly ARR trends" and an AI agent explaining "what drove growth last quarter?" both ride the same optimized path for freshness and latency.

Core and edge: a healthy division of labor

The most successful architectures balance central **authority** with local **adaptability**.

- **The core** — the "golden" layer — holds authoritative metric definitions, certified measures, standard dimensions and enterprise-wide policies. Core semantics change slowly through formal review and impact analysis.
- **The edge** — per team, application or agent — is seeded from the core and enhanced with additional knowledge from real-time interactions. For example, a sales team extends customer metrics with unique territory definitions, a product team experiments with funnel metrics and a region adds local synonyms and formatting.

Furthermore, useful Edge knowledge is then **reviewed and promoted** back to the core, so the enterprise layer evolves without drifting into chaos. This closes the loop: *author anywhere, govern centrally; learn locally, promote globally*.

Key principle: Author anywhere, govern centrally. Learn locally, promote globally.

The path to living semantics

The long-term vision is a semantic layer that **observes, adapts and improves** — capturing new concepts, evolving definitions and becoming more precise through usage. Platform-native placement makes this possible.

- Metrics learn clearer explanations from real usage
- Dimensions suggest new hierarchies from query patterns
- Synonyms evolve with business language
- Performance strategies adapt to live workloads
- Quality checks detect anomalies and drift automatically

This isn't a distant future — it's the natural result of treating semantics as managed, observable assets: curated in the **core**, extended at the **edge** and continuously improved through use.

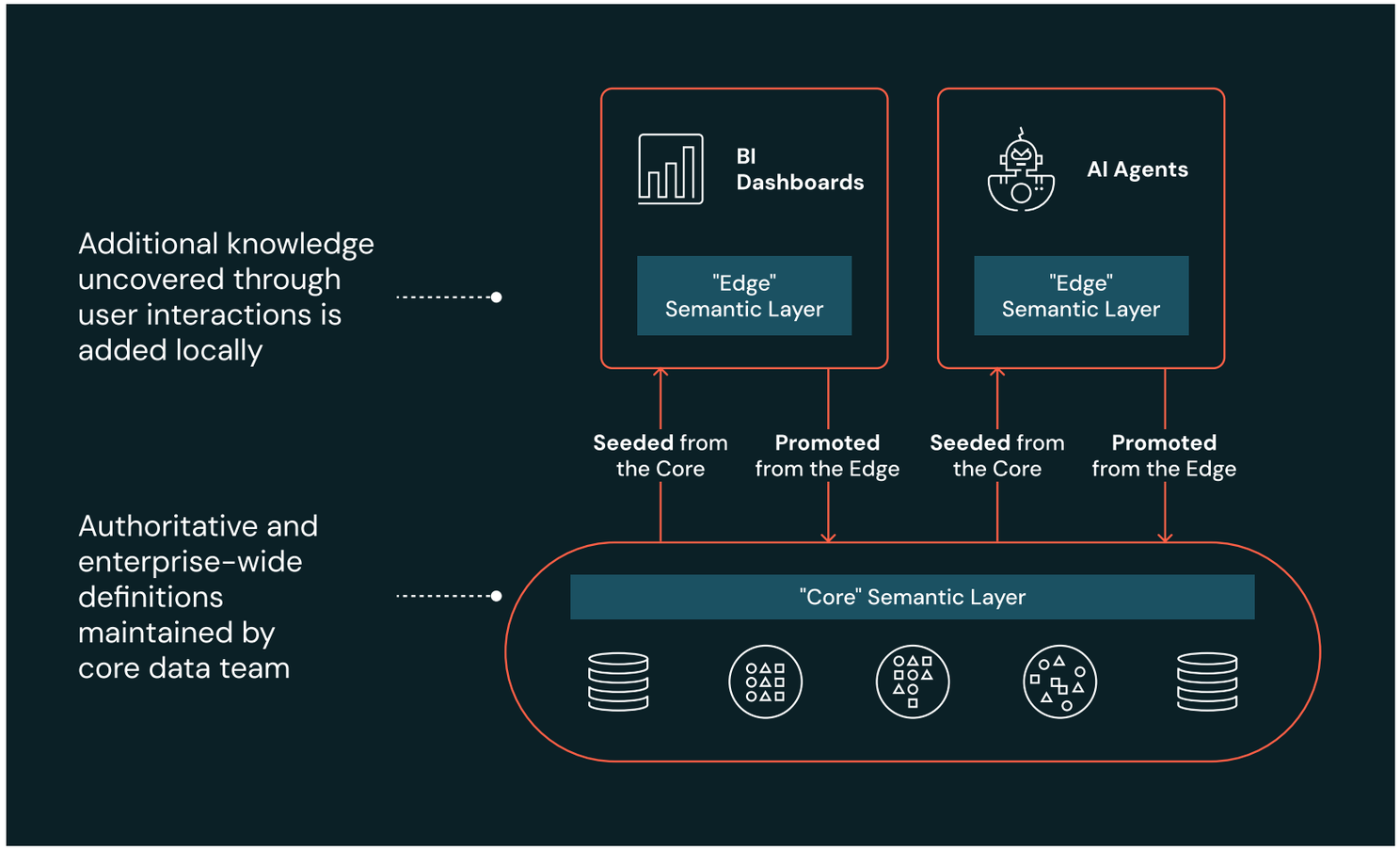


Figure 4. A living semantic system. Core definitions seed Edge semantics. Additional knowledge is added on the edge and then promoted back to the core where appropriate.

The platform approach doesn't just fix today's inconsistencies — it lays the foundation for tomorrow's intelligent analytics. As AI becomes central to how organizations understand data, the semantic layer becomes the shared intelligence that makes that understanding reliable, consistent and continuously improving.

Unity Catalog: Platform Native Semantics in Practice

Moving semantics into the data platform is no longer just an architectural choice — it's the foundation for reliable, scalable and governed business meaning in the age of AI. Unity Catalog is Databricks' unified governance layer and catalog for data and AI. It centralizes fine-grained access control, auditing, lineage, business semantics and discovery across workspaces and data types.

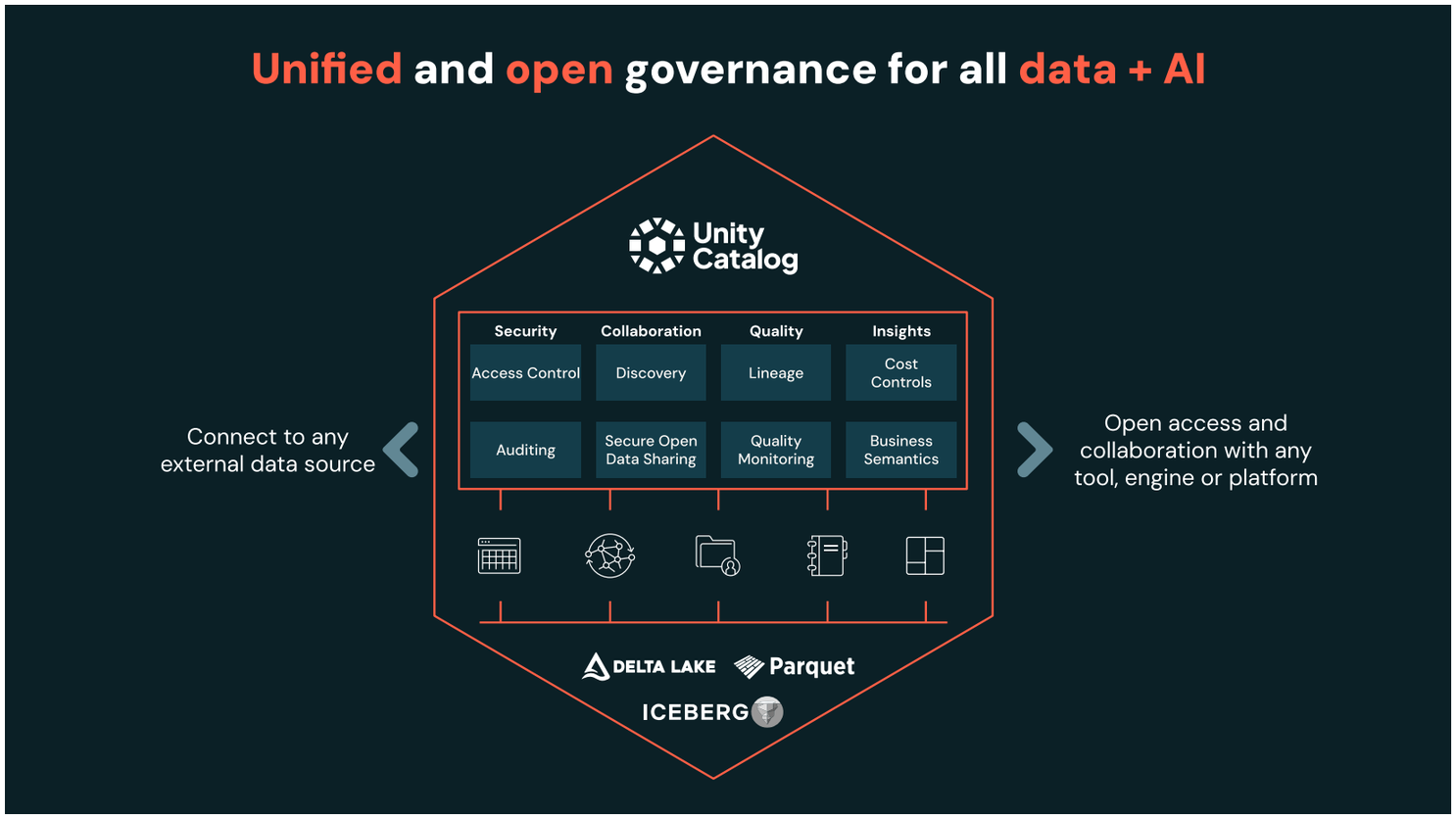


Figure 5. Unity Catalog is the central governance layer of the Databricks Data Intelligence Platform and the foundation for business semantics.

Unity Catalog embodies the principle of data platform native semantics. By centralizing definitions, policies and lineage next to the data itself, Unity Catalog transforms metrics, dimensions and business context into platform-native assets that are **discoverable, reusable and secure**.

The result is a semantic layer that's not an afterthought or a bolt-on, but a living part of your core data infrastructure — serving every dashboard, notebook and AI agent with the same consistent truth.

Unity Catalog Business Semantics operationalizes this promise. It provides two complementary pillars:

- **Metric views:** Platform-governed measures and dimensions available through SQL
- **Agent metadata:** Supplies extra business context and interpretability for both human users and AI agents

Together, they enable organizations to move beyond the limitations of tool-bound models and fragmented definitions, toward a unified, governed and continually evolving layer of business meaning — ready for every analytical and AI workload.

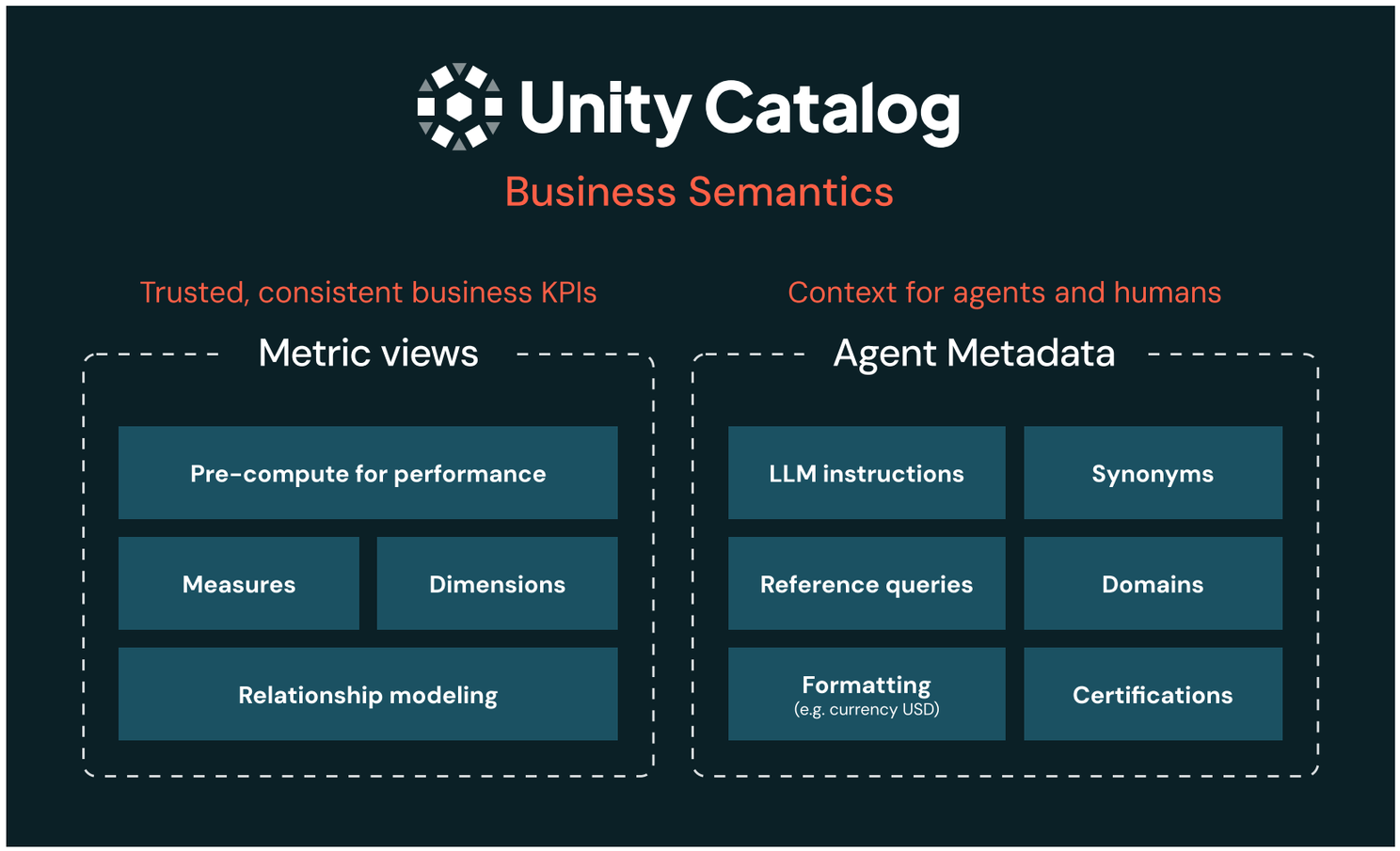


Figure 6. Unity Catalog Business Semantics is the combination of metric views and agent metadata.

Metric views: Crafting business logic for every surface

When consistent business meaning matters, metric views in Unity Catalog provide the language bridge between raw fact tables and every analytic or AI surface. They do this by distilling four foundational constructs: dimensions, measures, joins and filters. These aren't just technical components — together they encode the way your business thinks, segments and measures success, directly in the data platform.

DIMENSIONS: THE AXES OF ANALYSIS

Dimensions are the "who," "what," "where," and "when" by which performance is evaluated. Defined as categorical or temporal attributes, dimensions can represent regions, customer segments, product families, fiscal periods, cohorts and much more. In a SaaS company, dimensions commonly include:

- Customer segment (e.g., "enterprise," "SMB")
- Subscription type (e.g., "annual," "monthly")
- Product family (e.g., "platform," "add-ons")
- Order month (e.g., DATE_TRUNC('month', order_date))

Every measure — ARR, NRR, churn rate — can be grouped or filtered by these dimensions, enabling flexible analysis without redefining business logic each time.

MEASURES: PRECISE, REUSABLE QUANTIFICATION

Measures are the central values that quantify business outcomes, expressed as aggregate functions (e.g., SUM, COUNT, AVG, ratios, rolling windows). Measures are authored once, independent of grouping, so a definition like "net revenue retention" (NRR) carries the same meaning whether viewed by product, geography or time period. For example, key SaaS measures include:

- ARR run-rate: SUM(bookings_amount) * 12 / 3, filtered for active contracts
- NRR: SUM(expansion_amount) - SUM(churn_amount), filtered for active accounts
- Revenue churn rate: SUM(churn_amount) / NULLIF(SUM(starting_arr), 0)

JOINS: CONTEXT FROM ACROSS THE BUSINESS

A real business answer typically draws on more than one source. Metric views support both star and snowflake schema joins, letting you enrich primary facts (subscriptions, orders) with related dimensions (customer regions, product categories, contract types) and even traverse into subdimensions (such as team within region or product hierarchy).

For example, starting from a fact_subscriptions table, you might join to customers for segmentation and products for family/category, then further into regions via customer geography. These relationships are declared transparently in YAML, making lineage and governance visible.

FILTERS: SCOPED, SEMANTIC PRECISION

Filters let you encode business rules — such as “active contracts only,” “bookings from last 90 days” or “exclude test customers” — directly within the metric view. Each filter becomes a durable part of the metric’s definition, not an ad hoc WHERE clause in a dashboard.

AUTHORING A SAAS METRIC VIEW — COMPREHENSIVE EXAMPLE

Let's bring these concepts together using an illustrative YAML example for our SaaS scenario:

```yaml
version: 0.1

source: prod.sales.fact_subscriptions
filter: source.order_date >= '2025-01-01' AND source.subscription_status = 'active'

joins:
  - name: customer
    source: prod.sales.customers
    on: source.customer_id = customer.customer_id

  - name: product
    source: prod.sales.products
    on: source.product_id = product.product_id

dimensions:
  - name: Customer Segment
    expr: customer.segment

  - name: Subscription Type
    expr: subscription_type

  - name: Product Family
    expr: product.family

  - name: Order Month
    expr: DATE_TRUNC('month', order_date)

measures:
  - name: ARR Run-Rate
    expr: SUM(bookings_amount) * 12 / 3

  - name: NRR
    expr: SUM(expansion_amount) - SUM(churn_amount)

  - name: Revenue Churn Rate
    expr: SUM(churn_amount) / NULLIF(SUM(starting_arr), 0)
```

How it works in practice

- When a user asks for "ARR run-rate by customer segment in Q1," the system leverages defined dimensions and measure logic, executes the correct joins and applies the filters for active subscriptions and time window — producing governed, consistent results.
- Downstream — whether in dashboards, SQL, notebooks or AI/BI Genie spaces — users leverage the SQL-based `MEASURE()` function, choosing any mix of dimensions for instant, reliable answers.

In our SaaS scenario, let's look at how this would be queried in SQL:

SQL

```
SELECT
  "Customer Segment",
  "Subscription Type",
  "Cohort Month",
  "Order Month",
  MEASURE("ARR Run-Rate"),
  MEASURE("Rolling 30-Day Bookings"),
  MEASURE("Conversion Rate by Cohort"),
  MEASURE("YoY ARR Growth")
FROM
  prod.sales.fact_subscriptions_metrics_view
GROUP BY
  "Customer Segment",
  "Subscription Type",
  "Cohort Month",
  "Order Month";
```

Note: When you query a metric view, all filters defined in its YAML — such as restricting to active subscriptions or records after a certain date — are automatically and consistently enforced by the platform for every measure. This means that users and downstream tools always receive results that already reflect these business-specific constraints, without needing to remember or reapply them in their own SQL.

Why it matters

By standardizing the way dimensions, measures, joins and filters are described and enforced, metric views ensure that every business surface, every team and every AI agent operates from the same data vocabulary. Questions asked anywhere — "How's our NRR trending by subscription type?" — are always answered with the same trusted, governed logic.

ADVANCED SCENARIOS: WINDOWING, COHORTS AND TREND ANALYSES

Metric views make it easy to answer complex, time-sensitive business questions directly in the platform. For example, in our SaaS business, rolling window measures let you track momentum — such as trailing 30-day bookings or churn — so everyone sees the latest pulse of the business. Cohort analysis enables you to group and compare customer behavior by signup month, revealing retention or conversion patterns over time. Period-over-period metrics, like year-over-year ARR growth, expose trends that drive strategic decisions. All these analytics are defined once — in a reusable semantic layer — so every dashboard, query or AI agent relies on the same trustworthy logic.

Illustrative YAML (fragment):

```
# Rolling 30-day bookings measure
- name: Rolling 30-Day Bookings
  expr: SUM(
    CASE WHEN order_date >= CURRENT_DATE - INTERVAL '30' DAY THEN bookings_amount ELSE 0 END
  )

# Cohort conversion rate measure
- name: Cohort Month
  expr: DATE_TRUNC('month', signup_date)

- name: Conversion Rate by Cohort
  expr: SUM(
    CASE WHEN converted = TRUE THEN 1 ELSE 0 END
  ) / COUNT(DISTINCT customer_id)

# Day-over-day growth measure
 - name: previous_day_sales
    expr: SUM(o_totalprice)
    window:
      - order: date
        range: trailing 1 day
        semiadditive: last
  - name: current_day_sales
    expr: SUM(o_totalprice)
    window:
      - order: date
        range: current
        semiadditive: last
  - name: day_over_day_growth
    expr: (MEASURE(current_day_sales) - MEASURE(previous_day_sales)) / MEASURE(previous_day_
sales) * 100
```

These patterns make it straightforward to answer dynamic questions — from tracking recent performance, to monitoring longitudinal customer journeys, to benchmarking growth across periods — using governed, consistent business logic in every tool.

GOVERNANCE: CERTIFICATION, LINEAGE AND POLICY

Moving business semantics into Unity Catalog fundamentally transforms governance — from a patchwork of documentation and manual checks into an automated, auditable, always-on safeguard for your most critical business logic. With metric views, everything the business relies on — ARR definitions, retention formulas and revenue attribution — becomes a first-class, governed object, sitting right alongside your base fact tables and dimensions in the catalog itself.

Ownership and stewardship are explicit: every metric view can be assigned an owner and enriched with detailed descriptions that ensure the next analyst, engineer or executive understands its intent and logic. Certification is more than a badge. It's a signal both to people and to AI agents that this metric is approved, trusted and ready for use across the enterprise. Only certified metric views flow into key analytical surfaces and agent responses, closing the loop from analytic rigor to operational outcomes.

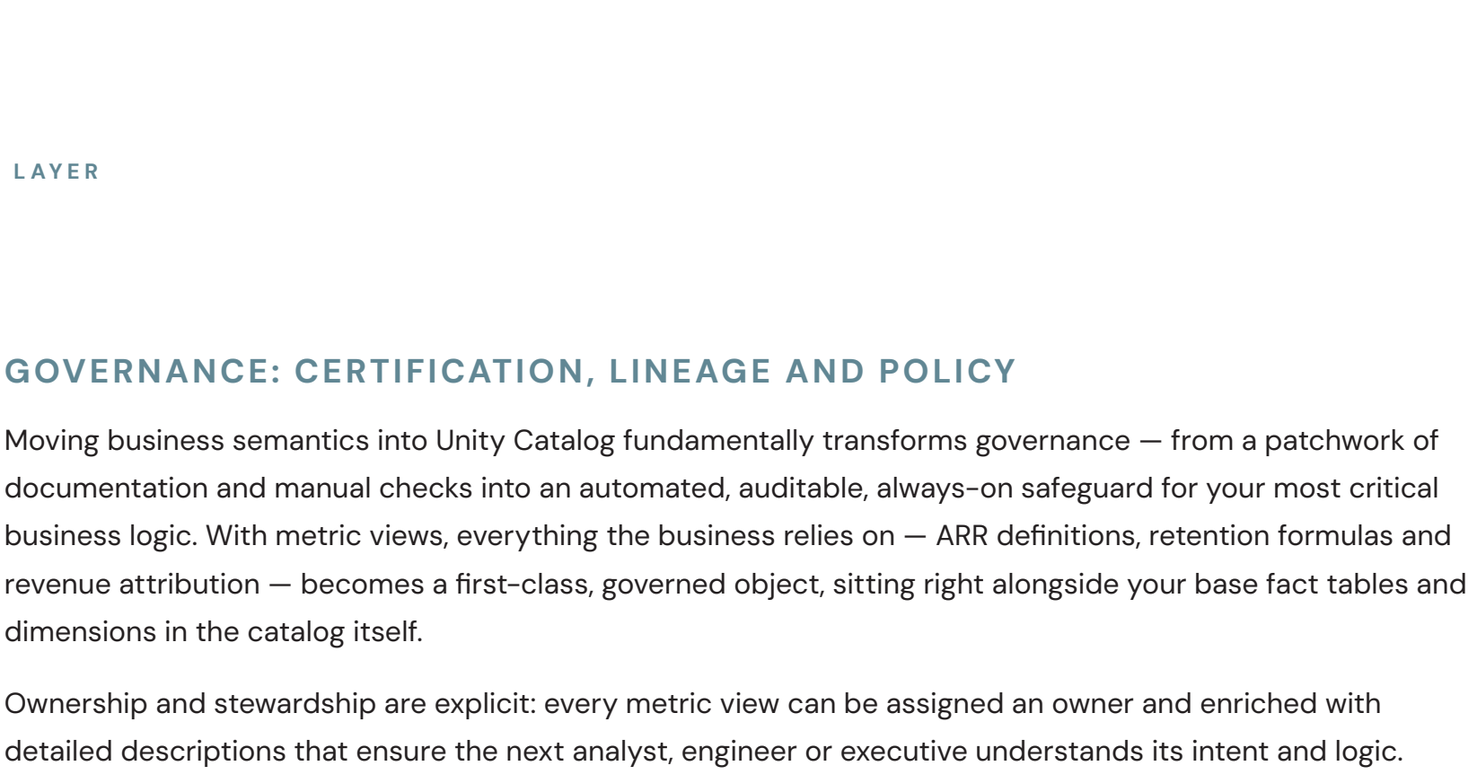


Figure 7. Screenshot of "Certified" metric view in Unity Catalog.

Automatic lineage is deeply embedded in Unity Catalog. Changes are captured at the definition level, and impact analysis is always just a click away. If finance updates how NRR is calculated, every lineage-aware tool instantly reflects this change, revealing downstream BI dashboards, notebooks and Genie spaces that will be affected. This visibility makes change management safe — no more blind spots or downstream breakage when updating business logic.

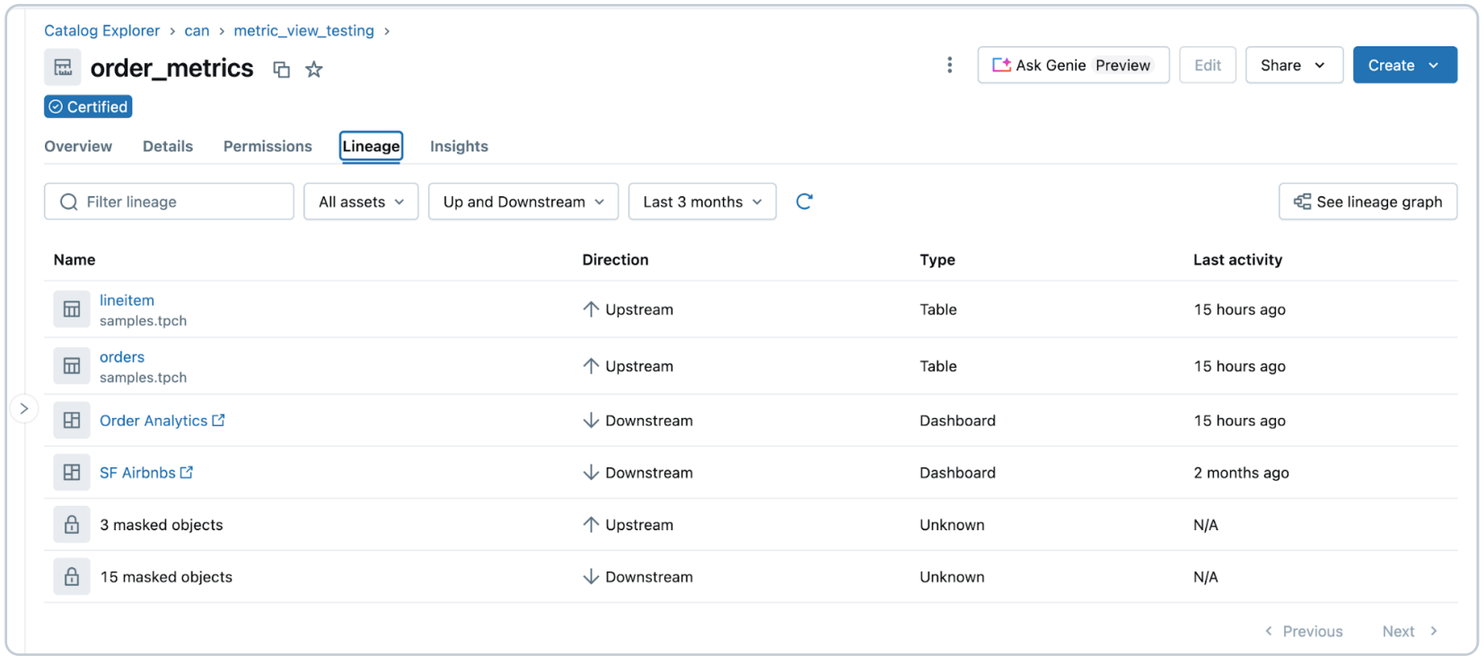


Figure 8. Screenshot of metric view lineage, automatically captured in Unity Catalog.

Security and policy are not afterthoughts, but fabric. The same fine-grained row- and column-level access controls set on your source data — think: only seeing customer details for your assigned region or masking sensitive contract values — are honored and enforced automatically within every metric view. Policies travel with the data, so you never have to worry that a derived calculation might expose what shouldn't be visible. When an executive asks a question via Genie or an AI/BI Dashboard refreshes, the platform quietly enforces every rule, every time.

Discovery is modern and intuitive. Tags let you organize metric views by domain (sales, finance, product), usage (certified, draft) or team, making it easy for business users and curators to navigate and activate the knowledge they need. Every metric view is searchable, auditable and programmable — just like any asset living in Unity Catalog.

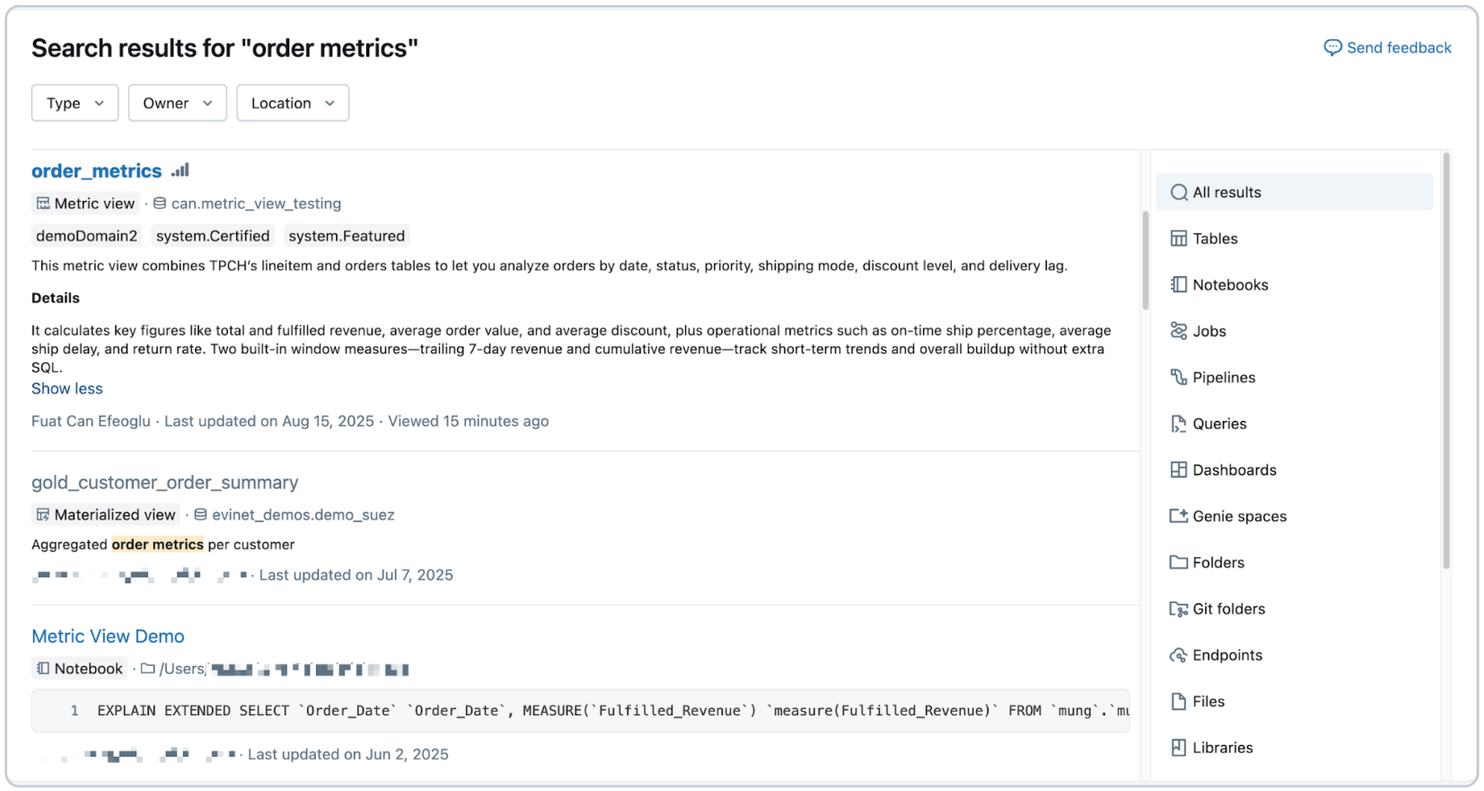


Figure 9. Discoverable metric view within the Databricks workspace.

Why it matters

This is the promise of platform semantics: governance that's no longer bolted on or scattered across disconnected tools, but a living guarantee that the numbers you see, the definitions you trust and the policies you enforce are always in sync, always up to date and always one permission check away. Decisions get safer, audits get faster and your most valuable business meaning gets the same protection as your most sensitive data.

PERFORMANCE AS A SHARED SERVICE: MATERIALIZATION

When business semantics live in the platform, acceleration becomes a shared service. Metric views support materialization strategies that precompute results at both detail and aggregate levels, **ensuring that critical key performance indicators (KPIs) respond instantly**, whether they're queried from dashboards, notebooks or AI agents asking complex questions in plain English.

Materialization works by creating two types of optimized storage: baseline materializations that cache filtered, joined source data with all necessary columns, and aggregated materializations that precompute specific measure-dimension combinations. The platform intelligently routes queries to the best available materialization — falling back to baseline or source data when needed — while maintaining the same governed definitions and security policies throughout.

Because materialized metric views are Unity Catalog objects, they inherit all platform governance automatically. Permissions, lineage and audit trails flow seamlessly from source tables through every materialized layer. When source data changes, refreshes cascade through the materialization hierarchy on schedules you control, keeping performance and consistency in balance.

Materialization in practice: SaaS KPI acceleration YAML example (fragment):

```
dimensions:
  - name: Customer Segment
    expr: customer.segment
  - name: Order Month
    expr: DATE_TRUNC('month', order_date)

measures:
  - name: ARR Run-Rate
    expr: SUM(bookings_amount) * 12 / 3

materialization:
  schedule: every 1 day
  mode: relaxed
  materialized_views:
    - name: baseline_cache
      type: unaggregated
    - name: arr_by_segment_month
      type: aggregated
      dimensions: [Customer Segment, Order Month]
      measures: [ARR Run-Rate]
```

With this configuration, Databricks maintains both a baseline cache of all subscription data and a pre-aggregated view of ARR by customer segment and month. Queries for ARR trends automatically use the fastest path available, while ad-hoc questions that slice by other dimensions fall back gracefully to the baseline or source — all transparently, all governed, all fast.

Agent metadata: Teaching AI to understand your business

Business metrics alone aren't enough to make AI useful. A table might define revenue_churn_rate precisely, but that doesn't mean an AI agent knows what a user means when they ask, "Why are VIPs churning more in Q4?" The math might be correct, but the meaning? The explanation? The context? That's what agents need to get right.

Agent metadata provides this missing layer of context. It describes how humans talk about data — so that AI can interpret questions, reason safely and explain answers faithfully. It lives in Unity Catalog as a governed source of truth, but it also evolves locally through usage and feedback. Together with metric views, Agent metadata turns Unity Catalog into **a machine-readable map of business meaning** — ready to be used by AI agents and traditional tools alike.

GROUNDING AI IN UNITY CATALOG

AI agents are only as reliable as the semantic foundation they're built on. Without structured context, even the most advanced models can misinterpret questions, overlook critical filters or fabricate explanations that deviate from business truth.

Unity Catalog provides the authoritative starting point — a platform-native semantic layer that any AI agent can rely on to ground its reasoning. This includes:

- **Table and column descriptions** that clarify meaning and purpose
- **Tags and glossary terms** that associate data assets with business domains
- **Synonyms and acronyms** that map colloquial terms to canonical fields
- **Formatting rules** that define how values should be displayed (e.g., rounding, percent, currency)
- **Policy-aware metadata** that enforces row- and column-level security automatically
- **Trusted dimensions, measures, joins and filters** exposed through metric views

This structured metadata doesn't just help analysts browse a catalog — it provides machine-readable context that AI agents can use to answer questions accurately, generate safe and performant queries and explain their results in business-friendly terms. Because Unity Catalog is deeply integrated into the data platform, its semantics aren't bolted on — they're enforced by construction. That means agents built on Unity Catalog inherit the same guarantees around access, lineage, certification and performance that govern every other asset in the system.

This architecture allows you to define business meaning once and project it outward to every consuming surface — whether that's a notebook, a BI dashboard or an AI agent. Databricks AI/BI Dashboards and agents like Genie leverage this foundation today, but the design is intentionally general: any AI interface that speaks to data can use Unity Catalog as its semantic backbone.

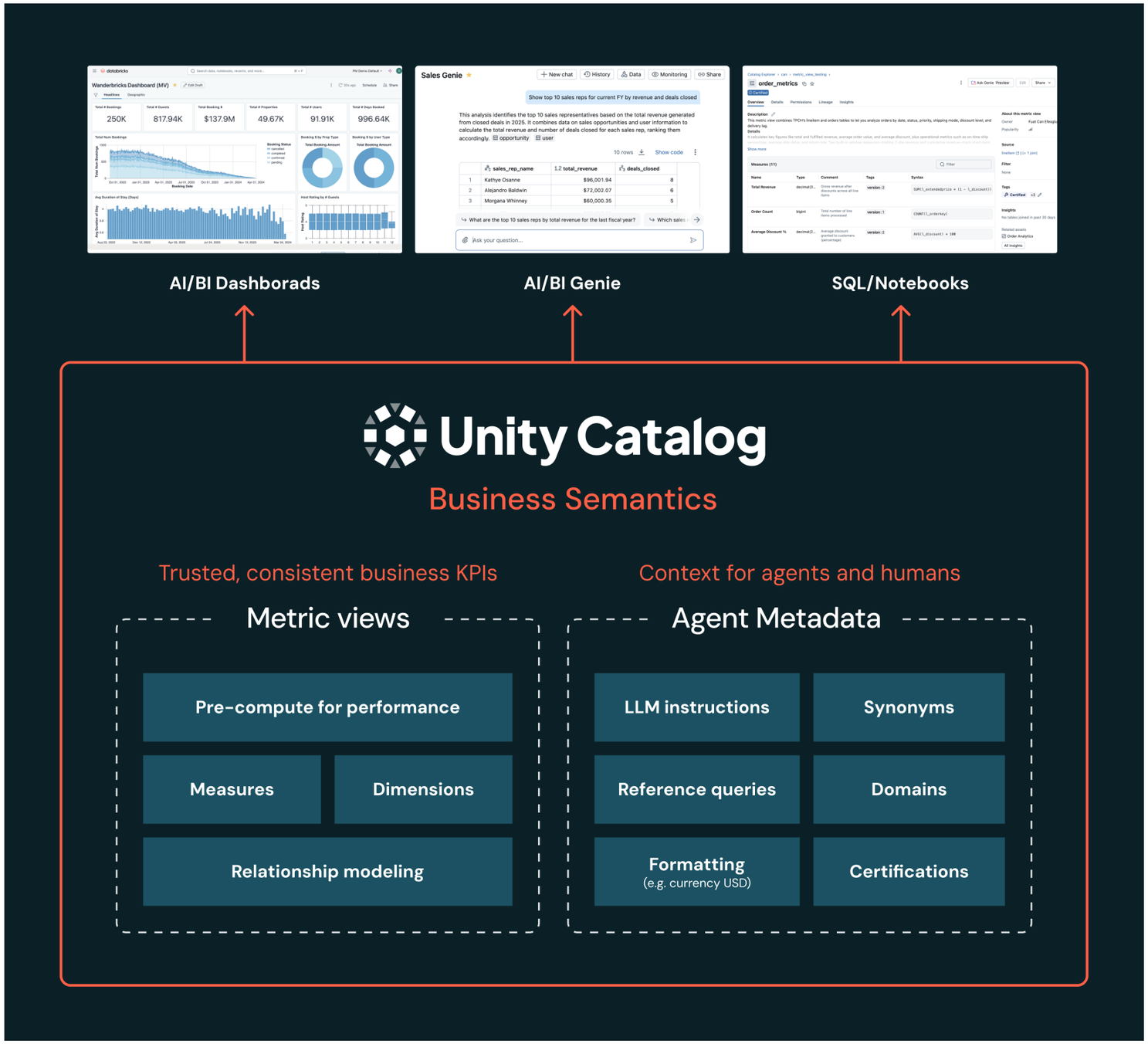


Figure 10. Unity Catalog Business Semantics (metric views and agent metadata) provides the necessary foundation for BI tools, AI agents and other developer surfaces.

By centralizing metadata, governance and semantic logic in the platform itself, organizations ensure that all AI-powered tools work from the same language — accurate, consistent and aligned with the business's actual thinking.

GENIE KNOWLEDGE STORES: LEARNING AT THE EDGE

While Unity Catalog defines the global truth, every Genie space maintains its own local knowledge store. These stores act as team-level or workspace-level extensions of the core semantics. They are:

- **Seeded from Unity Catalog**, inheriting its certified metrics, dimensions, joins, display rules and synonyms
- **Expandable by authors**, who can tailor the agent's understanding to the specific datasets, use cases or business language of their team

From there, **authors can enrich the knowledge store** with:

- Natural language instructions that refine interpretation or scope
- Example SQL queries that serve as teaching examples for AI
- Additional synonyms, descriptions or display rules not yet captured in Unity Catalog
- Custom metrics or filters specific to their domain
- Adjustments to phrasing or business logic to reflect local context

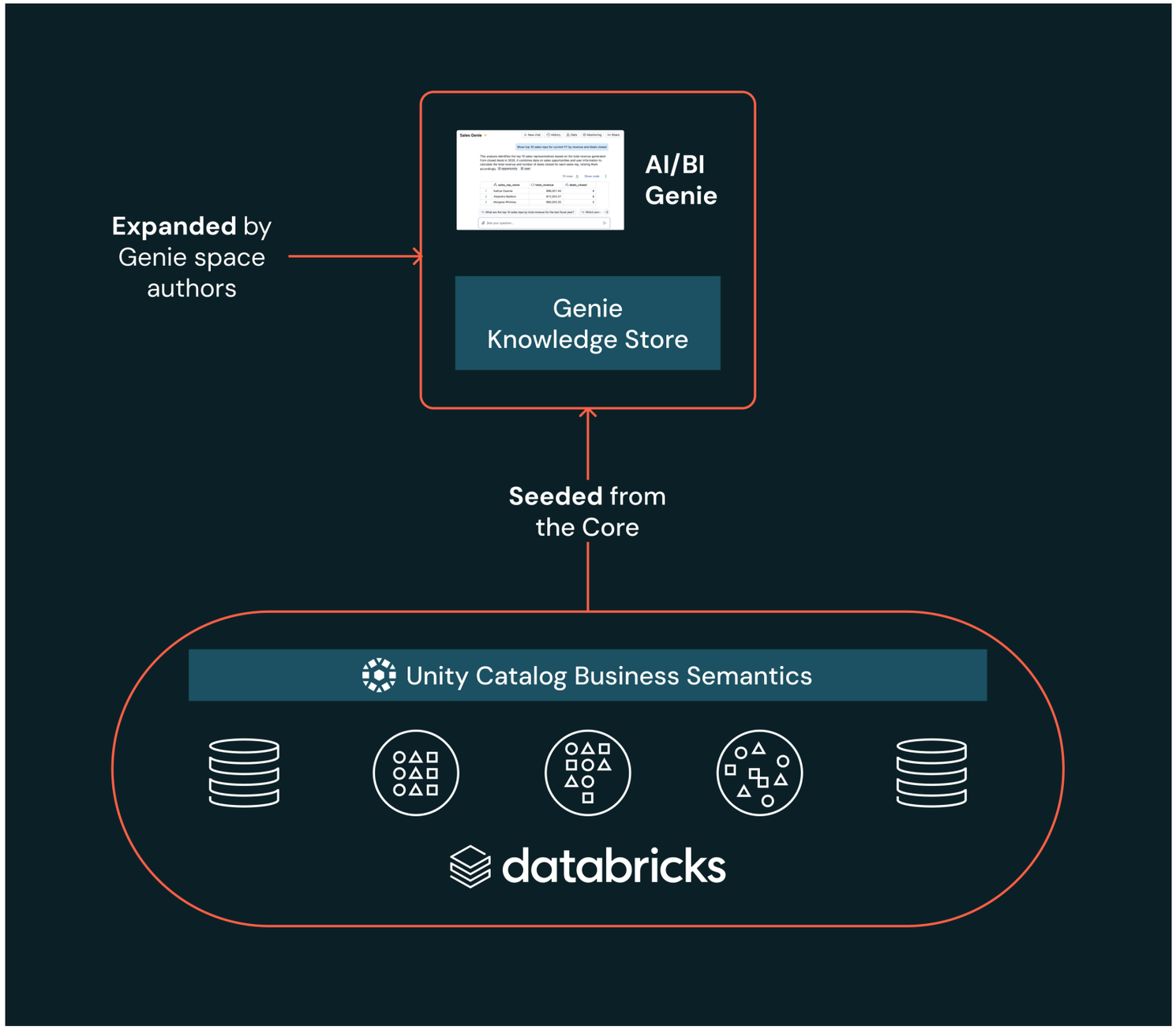


Figure 11. Genie knowledge stores are seeded with semantics from Unity Catalog and enriched by Genie space authors to provide additional context to the AI.

These additions don't replace the core — they build on top of it, enabling AI to reflect both centralized governance and local nuance. And because they're structured and auditable, they don't drift into prompt engineering or ephemeral hacks. They're real, inspectable metadata that align with how Databricks governs all assets.

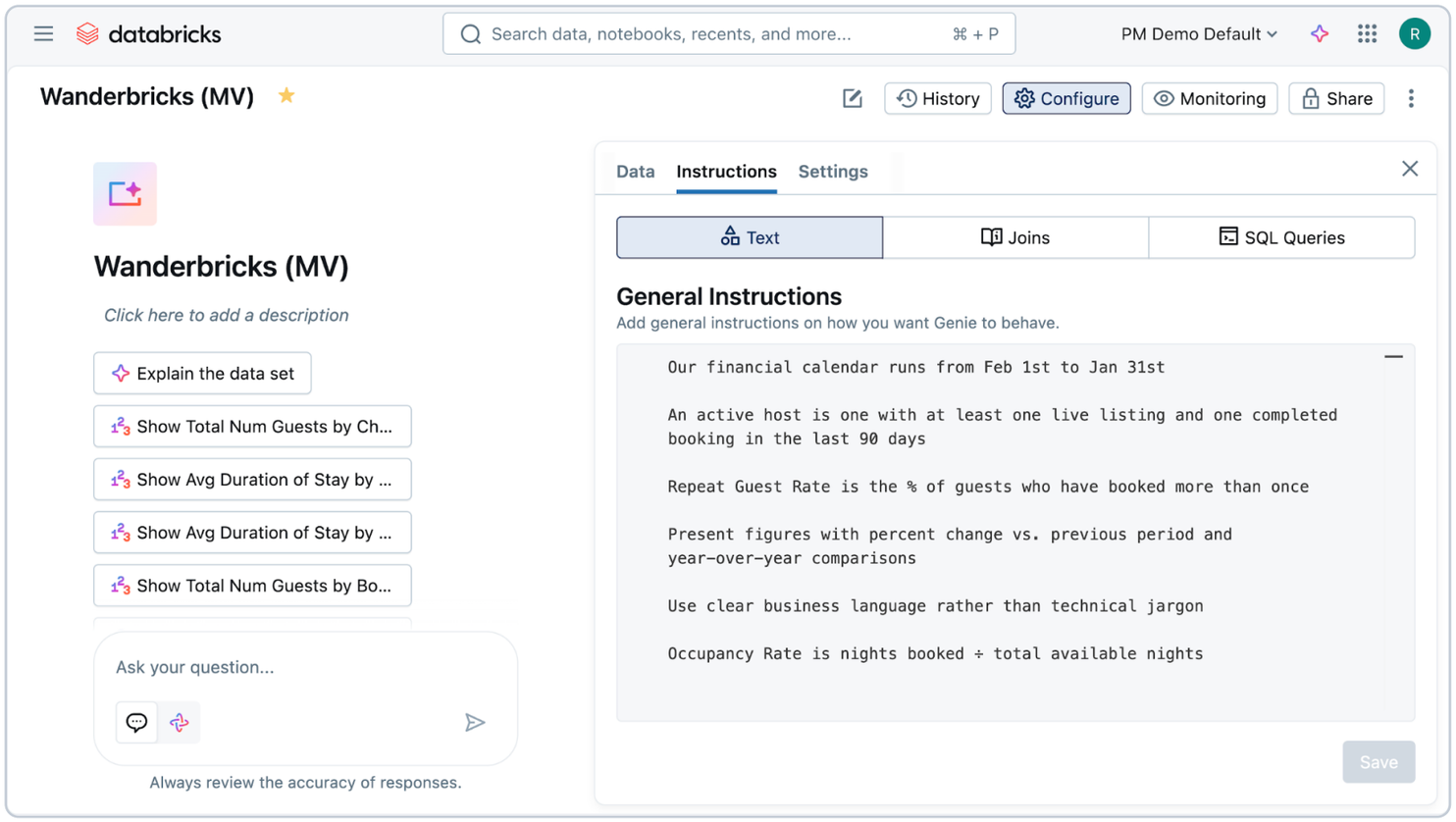


Figure 12. Screenshot of natural language instructions added to the local Genie knowledge store to improve context.

KNOWLEDGE MINING AND EXTRACTION: A CONTINUOUS-LEARNING AI

In traditional semantic systems, updates happen through quarterly audits or slow consensus cycles. Genie introduces something different: a live learning loop that refines semantics continuously, based on real-world usage.

This process is called knowledge mining and extraction, and it includes both passive pattern recognition and active concept learning.

A. Passive learning from usage

As users write queries — whether in SQL or natural language — Genie performs knowledge mining to identify patterns. For example, if many users write similar queries for "high spend customers," Genie proposes turning that into a reusable concept with sample queries and a definition. These become suggestions in the knowledge store, ready for author approval.

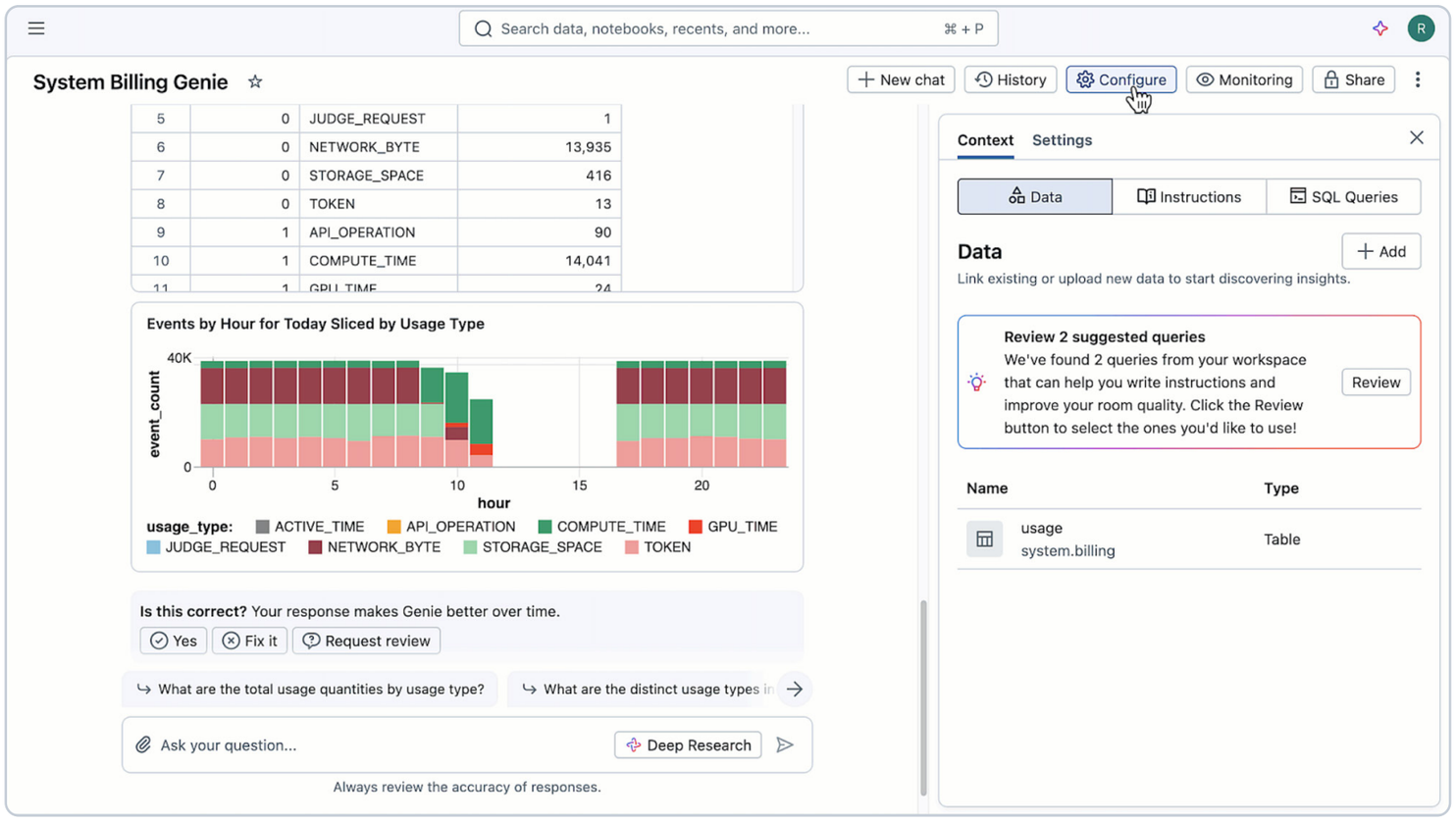


Figure 13. Screenshot of Genie knowledge mining in action. Here, Genie suggests two queries that can be reviewed and turned into new instructions.

B. Active learning through dialogue

Sometimes a user uses an unfamiliar term — like "platinum customers." Instead of failing, Genie asks what it means. When the user replies "customers with NPS > 8 and lifetime spend > 5000," Genie translates that into structured metadata: a new measure, filter or semantic concept. This Knowledge Extraction becomes a proposed snippet in the knowledge store — creating new semantic knowledge in real time.

These learned concepts can span:

- New filters or cohort definitions
- Composite measures
- Custom groupings
- Clarified synonyms
- Safe SQL templates for repeatable questions

This continuous-learning loop ensures that agents don't become stale. They evolve with the business, capturing the latest phrasing, edge cases and analytical patterns — without requiring constant rebuilds.

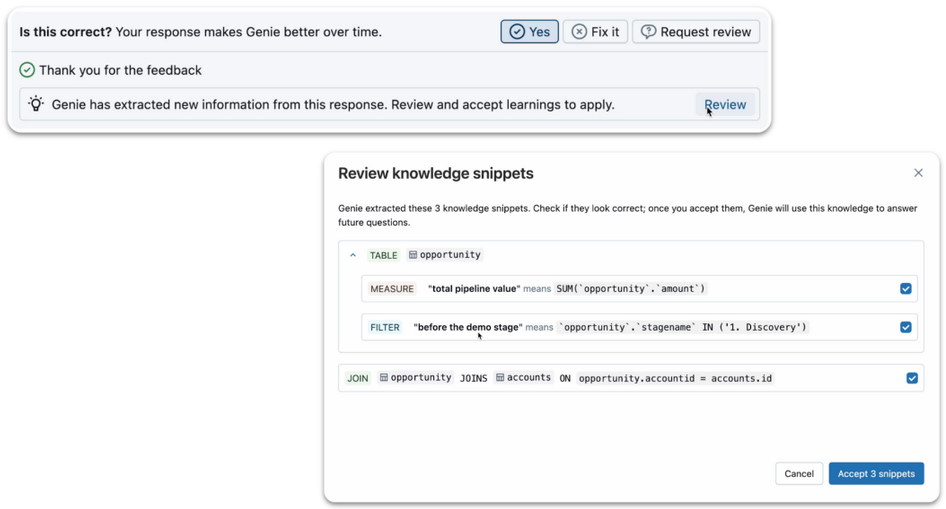


Figure 14. Screenshot of Genie Knowledge Extraction in action. Here, Genie has dynamically learned the concept of "Total Pipeline Value" based on clarification provided by the end-user. Genie then presents these as new knowledge snippets for the author to add to the local knowledge store.

THE FEEDBACK LOOP: PROMOTING KNOWLEDGE TO THE CORE

Perhaps the most powerful part of this system is what happens next: **edge knowledge doesn't stay at the edge**.

Any new concept or clarification created in an A/BI Dashboard or Genie knowledge store — whether manually authored or learned through usage — can be promoted back to Unity Catalog. That **turns localized insight into platform-wide knowledge**.

This is possible in two key ways:

- **From the local Genie knowledge store:** New synonyms, display rules, filters or measures created in a Genie space can be pushed upstream as governed metadata. Once approved, they become reusable across other agents, dashboards and tools.
- **From any AI/BI Dashboard dataset:** Similarly, if an analyst creates a derived dataset or logic in an AI/BI Dashboard, it can be published as a new metric view, adding to the organization's shared semantic layer.

This feedback loop is a critical differentiator in the Databricks architecture. Other tools lock business knowledge in artifacts — hidden in a dashboard or semantic model, ephemeral in an agent session or brittle in a query. But **with Unity Catalog Business Semantics, insights become infrastructure**. You don't just answer the question — you improve the system for the next one.

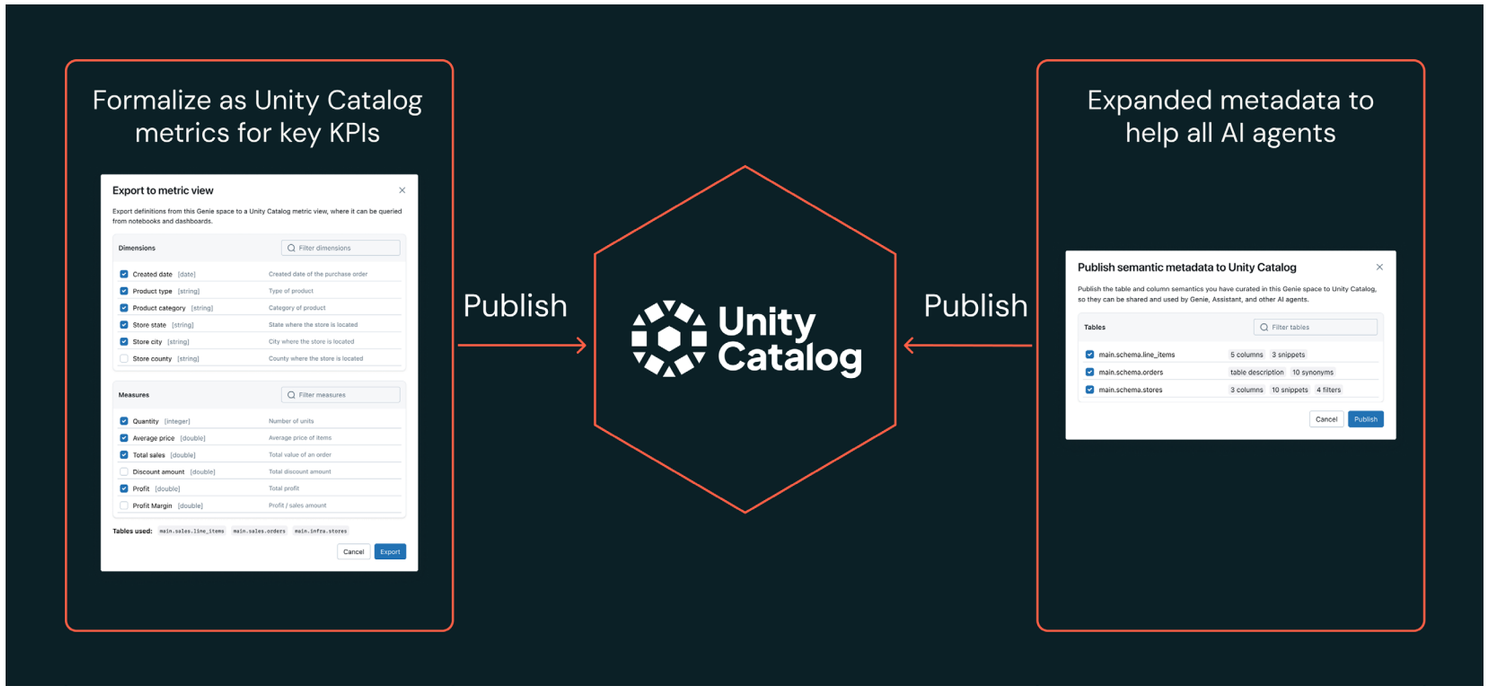


Figure 15. Screenshot of AI/BI Dashboards' ability to "publish" knowledge back to Unity Catalog as a new metric view. The image also shows publishing to Unity Catalog from a Genie knowledge store.

THE COMPLETE SYSTEM: SEMANTICS THAT THINK FOR THEMSELVES

Agent metadata completes the vision of a living semantic layer. It gives AI the context it needs to behave like a thoughtful, business-aware colleague — not just a SQL generator. And by combining platform grounding, edge learning, human curation and feedback promotion, it creates a self-evolving system:

- Platform: Unity Catalog defines the authoritative core
- Edge: Agents adapt to local needs via knowledge stores
- Learning: AI mines usage and dialogue to suggest new metadata
- Governance: Curators approve and promote back to the platform

This is how semantics scale in the era of AI: not by dictating from the center, but by enabling a system that listens, learns and improves — every day, in every interaction.

Where To Learn More About Unity Catalog Business Semantics

As we've seen throughout this book, the durable solution for analytics and AI is to manage semantics in the data platform and project them outward to every tool and user. Unity Catalog is the foundation for that approach in Databricks. The following resources will guide you if you want to deepen your knowledge and begin applying these concepts in practice.

Foundations: Unity Catalog and governed semantics

Unity Catalog is where semantics live alongside your data, policies and lineage. It provides the governance and consistency layer for everything you build.

- Unity Catalog product page
- Unity Catalog Business Semantics product page
- Unity Catalog documentation
- What's new with Unity Catalog (DAIS 2025)

Core semantics: Unity Catalog metric views

Metric views turn business logic — measures, dimensions, filters — into governed, reusable assets. Defined once in Unity Catalog, they can be used everywhere: SQL, Dashboards, Genie and applications.

- Metric views overview — what they are and where they fit
- Create a metric view tutorial — step-by-step via Catalog Explorer UI
- YAML reference — fields for measures, dimensions, filters and metadata
- Understanding your business with Unity Catalog metric views (DAIS 2025)

Product demos

See Unity Catalog Business Semantics and metric views in action

- Metric views overview video
- Unity Catalog metric views and Databricks SQL
- Interactive product tour

AI/BI and Unity Catalog Business Semantics

Start by watching the surfacing metric views in AI/BI demo video. Then discover more about AI/BI Dashboards and Genie.

Dashboards in Databricks build directly on Unity Catalog Business Semantics. This means every chart, filter and drill-down can be powered by governed metric views, ensuring consistent answers.

- AI/BI Dashboards product page
- AI/BI Dashboards documentation

Genie extends semantics into natural language. By grounding AI in Unity Catalog metadata and metric views, it allows business users to ask questions and receive accurate, contextual answers.

- Genie product page
- Genie documentation
- Genie generally available announcement blog

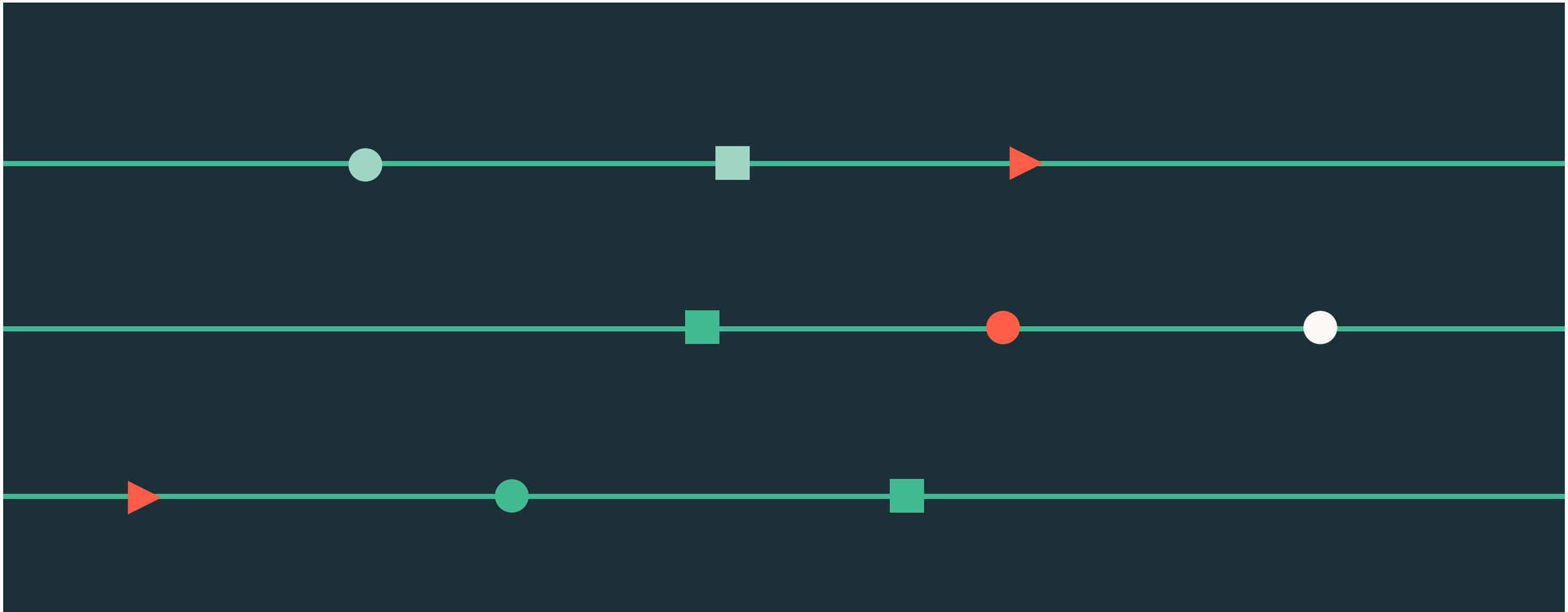
Training and hands-on learning

Databricks provides courses, demos and free trials so you can move from concept to practice quickly.

- Get Started With SQL Analytics and BI Course
- Databricks Fundamentals Course
- Databricks Free Edition
- Free Databricks Trial

In the end, a semantic layer isn't a product to admire — it's a practice to adopt. The resources above give you the how. The why is simple: when definitions live in Unity Catalog next to your data, policy and lineage, every surface — SQL editors and notebooks, BI dashboards and AI Agents like Genie — reuse the same governed truth.

Remember to start small and be visible: pick one business decision, define one metric and its key dimensions as a metric view, then use it in an AI/BI Dashboard and let Genie answer questions against it. As usage increases, learn more semantics locally, then promote globally. As you certify more logic, performance and trust become properties of the platform, not side projects. That's the arc of this book: author once, reuse everywhere, and make AI-ready analytics a habit — not a heroic effort.



About Databricks

Databricks is the data and AI company. More than 10,000 organizations worldwide — including Block, Comcast, Condé Nast, Rivian, Shell and over 60% of the Fortune 500 — rely on the Databricks Data Intelligence Platform to take control of their data and put it to work with AI. Databricks is headquartered in San Francisco, with offices around the globe, and was founded by the original creators of Lakehouse, Apache Spark™, Delta Lake and MLflow. To learn more, follow Databricks on LinkedIn, X and Facebook.

databricks