databricks

eBook

Big Book of Data Warehousing and BI

Contents

databricks

01

Introduction to the Data Intelligence Platform

databricks

Data is vital to the success of every company. As organizations increasingly rely on data, maintaining efficient data management and governance becomes more crucial. Addressing this, the Databricks Data Intelligence Platform enables effective data management, utilization and access to data and AI. Built on the lakehouse architecture, it merges the best features of data lakes and data warehouses, reducing costs and speeding up data and AI initiatives. The platform provides unified governance for data and AI, along with a versatile query engine for ETL, SQL, machine learning and BI.

The Databricks Data Intelligence Platform enables organizations to effectively manage, utilize and access all their data and AI. The platform — built on the lakehouse architecture, with a unified governance layer across data and AI and a single unified query engine that spans ETL, SQL, AI and BI — combines the best elements of data lakes and data warehouses to help reduce costs and deliver faster data and AI initiatives.

By combining generative AI with the unification benefits of a lakehouse, the Data Intelligence Platform offers a Data Intelligence Engine called DatabricksIQ that understands the unique semantics of your enterprise data. DatabricksIQ automatically analyzes all aspects of data, including its content, metadata and usage patterns (such as queries, reports and lineage). This comprehensive analysis enables the platform to continually learn, enhance and add new capabilities, optimizing data management and AI applications. Through this deep understanding of data, the Databricks Data Intelligence Platform enables:

Natural Language Access: Leveraging AI models, the Data Intelligence Platform enables working with data in natural language, tailored to each organization's jargon and acronyms. The platform observes how data is used in existing workloads to learn the organization's terms and offers a tailored natural language interface to all users — from nonexperts to data engineers.

Semantic Cataloging and Discovery: Generative AI can understand each organization's data model, metrics and KPIs to offer unparalleled discovery features or automatically identify discrepancies in how data is being used.

Automated Management and Optimization: AI models can optimize data layout, partitioning and indexing based on data usage, reducing the need for manual tuning and knob configuration.

Enhanced Governance and Privacy: Data Intelligence Platforms can

databricks

automatically detect, classify and prevent misuse of sensitive data, while simplifying management using natural language.

First-Class Support for AI Workloads: Data Intelligence Platforms can enhance any enterprise AI application by allowing it to connect to the relevant business data and leverage the semantics learned by the platform (metrics, KPIs, etc.) to deliver accurate results. AI application developers no longer have to "hack" intelligence together through brittle prompt engineering.
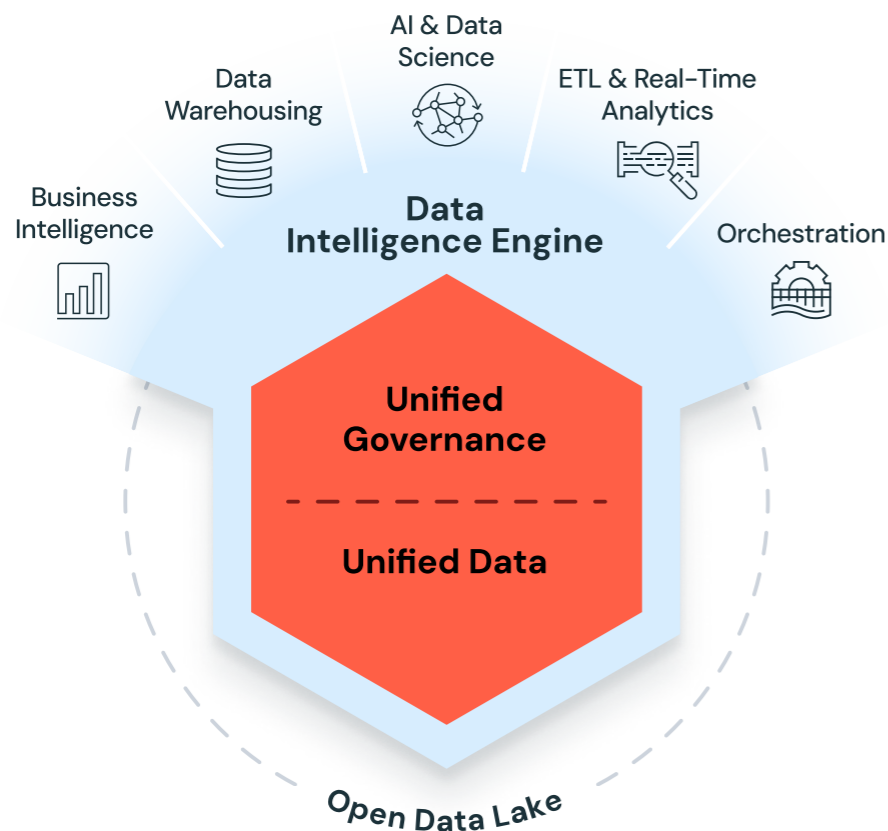
The Databricks Platform also simplifies the development of enterprise AI applications. The platform makes it easy for enterprises to build AI applications that understand their data. The Databricks Platform offers multiple capabilities to directly integrate enterprise data into AI systems, including:

- End-to-end RAG (retrieval augmented generation) to build high-quality conversational agents on your custom data

- Training custom models either from scratch on an organization's data, or by continued pretraining of existing models such as MPT and Llama 2, to further enhance AI applications with deep understanding of a target domain

- Efficient and secure serverless inference on your enterprise data, with unified governance and quality monitoring functionality

- End-to-end MLOps based on the popular MLflow open source project, with all produced data automatically actionable, tracked and monitorable in the lakehouse

This how-to reference guide showcases data warehousing best practices on the Databricks Data Intelligence Platform through end-to-end use cases from real-world examples. Discover how the platform helps businesses of all sizes translate raw data into actionable data using SQL — from data ingestion to data processing, AI and LLMs, analytics and BI. We'll arm you with reference architectures and code samples so you can explore all aspects of the data lifecycle on the Data Intelligence Platform.

To learn more about Databricks SQL, read our eBook Why the Data Lakehouse Is Your Next Data Warehouse.

AI & Data Science

Data Warehousing

ETL & Real-Time Analytics

Business Intelligence

Data Intelligence Engine

Orchestration

Unified Governance

— — — — — — —

Unified Data

Open Data Lake

databricks

Data Warehousing Built With Intelligence and Automation

databricks

Success in lakehouse-based data and AI initiatives hinges on a simplified data architecture, data quality and governance, and scalability and performance. These pillars collectively provide the foundation upon which organizations build their data strategies, guiding them through the intricate maze of modern data management and analytics.

Simplified Data Architecture

The data lakehouse architecture solves the problems of data silos while incorporating the capabilities of a data warehouse. The approach starts by centralizing data within a cloud-based data lake. This foundation, supported by **Delta Lake**, allows analytics and AI use cases to operate on a single data source — reducing storage expenses and streamlining data engineering. The lakehouse architecture integrates a unified governance and security structure with **Unity Catalog**, ensuring granular data control and timely access for respective teams.

The lakehouse architecture's holistic approach encompasses the entire data lifecycle, transformation and impact across various analytics and AI workloads. Because all workloads share the same data while adhering to uniform security and governance policies, you can feel confident knowing you can rely on the accuracy of the data. Functional silos diminish, paving the way for seamless collaboration and, consequently, enhanced productivity in delivering data products.

These are some of the additional benefits of a simplified data architecture:

- Built on open source and standards, a lakehouse removes data silos, simplifying the data landscape and facilitating data and AI operations

- One platform serves integration, storage, processing, governance, sharing, analytics and AI. It offers a unified approach for handling both structured and unstructured data, a complete view of data lineage and provenance, and a consolidated toolset for Python and SQL, notebooks, IDEs, batch, streaming and all primary cloud providers.

- Automated optimization for performance and storage delivers optimal TCO, setting performance benchmarks for data warehousing and AI — including advanced processes like large language models (LLMs)

databricks

Data Governance and Quality

Data, being fundamental to organizations, requires stringent quality and governance, especially with increasing volumes and variety. Organizations must prioritize accuracy, reliability and compliance to maximize their lakehouse's potential. Subpar data quality can distort insights, while weak governance can lead to regulatory and security lapses. The Databricks Data Intelligence Platform, with its Unity Catalog, addresses these issues, providing an integrated framework for managing and improving the quality of data across its lifecycle. With governance on the lakehouse architecture, you can:

- Discover, classify and consolidate data and AI assets from various platforms on any cloud, enhancing data exploration and insight extraction using natural language, all from a single access point

- Simplify access management through a unified interface, ensuring consistent and secure access across clouds and platforms, with enhanced fine-grained control and scalable low-code policies

- Harness AI to automate data and ML model monitoring, receive proactive alerts for issues, streamline debugging, and achieve holistic observability of your lakehouse operations using built-in system tables

- Efficiently share data and AI assets across clouds, regions and platforms using open source Delta Sharing in Unity Catalog, enabling secure collaboration and value creation without the need for complex processes or costly replication

Scalability and Performance

With growing data volumes, a lakehouse architecture distributes computing features, independent of storage, aiming to maintain consistent performance at optimal costs. The Databricks Data Intelligence Platform is designed for elasticity, allowing organizations to scale their data operations as needed. Scalability extends across various dimensions:

SERVERLESS

The Databricks Platform utilizes serverless cloud-based computing resources, enabling workloads to adjust and scale elastically based on the required computing capacity. Such dynamic resource allocation guarantees rapid data processing and analysis, even during peak demand.

CONCURRENCY

Leveraging serverless compute and AI-driven optimizations, the Databricks Platform facilitates concurrent data processing and query execution. This ensures that multiple users and teams can undertake analytics tasks concurrently without performance constraints.

STORAGE

The platform seamlessly integrates with data lakes, facilitating the cost-effective storage of extensive data volumes while ensuring data availability and reliability. It also optimizes data storage for enhanced performance, reducing storage expenses.

Scalability, though essential, is complemented by performance. In this regard, the Databricks Data Intelligence Platform stands out, offering a variety of AI-driven optimizations:

OPTIMIZED QUERY PROCESSING

The platform utilizes machine learning optimization techniques to accelerate query execution. It leverages automatic indexing, caching and predicate pushdown to ensure queries are processed efficiently, resulting in rapid insights.

AUTOSCALING

The Databricks Platform intelligently scales serverless resources to match your workloads, ensuring that you pay only for the compute you use, all while maintaining optimal query performance.

PHOTON

The new native massively parallel processing (MPP) engine on the Databricks Platform provides extremely fast query performance at low cost — from data ingestion, ETL, streaming, data science and interactive queries — directly on your data lake. Photon is compatible with Apache Spark™ APIs — no code changes and no lock-in.

DELTA LAKE

Delta Lake with Unity Catalog and Photon offers the best price/performance out of the box without manual tuning. The Databricks Platform uses AI models to solve common challenges with data storage, so you get faster performance without having to manually manage tables, even as they change over time.

- Predictive I/O for updates optimizes your query plans and data layout for peak performance, intelligently balancing read vs. write performance. So you can get more from your data without needing to decide between strategies like copy-on-write vs. merge-on-read.

- Liquid clustering delivers the performance of a well-tuned, well-partitioned table without the traditional headaches that come with partitioning, such as worrying about whether you can partition high-cardinality columns or expensive rewrites when changing partition columns. The result is lightning-fast, well-clustered tables with minimal configuration.

- Predictive optimization automatically optimizes your data for the best performance and price. It learns from your data usage patterns, builds a plan for the right optimizations to perform and then runs those optimizations on hyper-optimized serverless infrastructure.

Having established the foundation of the lakehouse architecture, it's pertinent to explore the delivery of data warehouse and analytics capabilities on Databricks with appropriate data structures and management functionalities facilitated by Databricks SQL (DB SQL).

databricks

Databricks SQL Serverless:
The best data warehouse for a lakehouse architecture

Databricks SQL was introduced to enhance data warehousing capabilities and offer premier SQL support on the Databricks Data Intelligence Platform. It streamlines SQL-based data transformation, exploration and analysis, catering to users of diverse technical backgrounds. From BI analysts and data architects to analytics engineers, its intuitive SQL interface facilitates queries and complex data operations without necessitating specialized programming. This broadened data access fosters an organization-centric culture, empowering more teams to base decisions on data insights.

Databricks SQL distinguishes itself with its ability to handle massive data sets with speed and efficiency. Utilizing Databricks' next-gen engine, Photon with AI-driven optimizations, ensures rapid data processing and analysis, notably decreasing query execution durations. High performance is crucial for organizations facing data challenges, guaranteeing insights from an extensive variety of data sets. Moreover, Databricks SQL champions collaboration, providing a workspace where data professionals can instantaneously share queries, outcomes and understandings. This shared setting promotes knowledge exchange and hastens resolution, allowing organizations to capitalize on their teams' collective intelligence.

Additionally, Databricks SQL incorporates advanced provisions for data governance, security and compliance, empowering organizations to uphold data quality, implement access restrictions, oversee data activities, safeguard sensitive data and adhere to regulatory standards. To sum up, Databricks SQL provides:

- **Faster Time to Insights**
  Use plain English to access data, and it will automatically create the SQL queries for you. This makes it faster to refine your queries and is available to everyone in the enterprise.

- **Best Price/Performance**
  Serverless compute combined with AI-optimized processing achieves top-tier performance and scale at lower costs, without the need for cloud infrastructure management

- **Unified Governance**
  Establish one unified governance layer across all data and AI assets no matter where they live

- **Reduce Complexity**
  Unify all your data, analytics and AI on one platform that supports SQL and Python, notebooks and IDEs, batch and streaming, and all major cloud providers

- **Rich Ecosystem**
  Utilize SQL and favorite tools such as Power BI, Tableau, dbt and Fivetran with Databricks for BI, data ingestion and transformation

databricks

Conclusion

The Databricks Data Intelligence Platform represents a significant advancement in the realm of data warehousing and analytics. It addresses the critical need for efficient data warehousing workloads in today's data-driven business landscape. The platform's simplified data architecture centralizes data with complete end-to-end data warehouse capabilities, leading to cost savings and speeding up time to turn raw data into actionable insights at scale — and unify batch and streaming. Moreover, it ensures data quality and governance through the Unity Catalog, enabling organizations to easily discover, secure and manage all their data with fine-grained governance with data lineage across clouds.

Scalability and performance are key foundational differentiators of the Databricks Platform, with serverless computing, concurrency support and optimized storage strategies. AI-driven optimizations enhance query processing, improve performance and optimize data for best performance and price, making data analysis faster and more cost-effective.

Databricks SQL further enhances the platform's capabilities by providing premier SQL support, facilitating data transformation and analysis for many users. It promotes collaboration, data governance and a rich ecosystem of tools, breaking down data silos and enabling your organization to harness the full potential of your data. Now, let's take a look at a few use cases for running your data warehousing and BI workloads on the Databricks Platform.

LEARN MORE

→ Why the Data Lakehouse Is Your Next Data Warehouse: 2nd Edition

→ What's new in Databricks SQL?

→ Introducing Lakehouse Federation Capabilities in Unity Catalog

→ Introducing AI Functions: Integrating Large Language Models with Databricks SQL

03

The Next Wave of
Business Intelligence

databricks

The Next Wave of Business Intelligence

Introduction to Databricks AI/BI

For Databricks customers, AI/BI represents a significant advancement in the area of BI by integrating AI directly into the analytics process. It allows users to ask questions in natural language and receive trusted, AI-generated insights. What sets this system apart is its native integration within the Databricks Data Intelligence Platform, which provides a unified environment for analytics, data management and governance. This tight integration offers distinct advantages over traditional, siloed BI systems that often operate separately from the underlying data platforms. By learning from the entire data estate, Databricks AI/BI delivers accurate insights that reflect both the complexity and the specific context of the organization's data.

Moving Beyond Traditional BI Systems

Traditional BI tools typically operate on static datasets or predefined reports, often housed in disconnected systems. This fragmented approach leads to inefficiencies, with challenges such as duplicated data and governance efforts and limited real-time analytics, resulting in slower time to insight. Databricks AI/BI circumvents these issues by embedding AI capabilities directly within the platform's native architecture.

Key benefits of this native integration include:

- A single, trusted source of data for all analytics, eliminating the need for maintaining separate BI systems

- Unified governance and security through Unity Catalog, ensuring consistent data access policies and robust compliance mechanisms

- High-performance querying and real-time insights are made possible through the seamless use of Databricks SQL

This approach streamlines operations, reduces data inconsistencies and allows organizations to scale their analytics capabilities without the typical technical bottlenecks seen in siloed BI systems.

databricks

AI/BI Dashboards: A New Paradigm for Data Analysts

Databricks AI/BI Dashboards provide a powerful environment where data analysts can leverage Databricks Assistant to streamline the development of analytical datasets, optimize SQL queries and leverage natural language to build effective visualizations. By integrating AI directly into the dashboard creation process, AI/BI enables analysts to develop complex dashboards faster and with greater accuracy. Whether it's improving SQL efficiency or automatically generating data visualizations from natural language queries, AI/BI Dashboards significantly reduce the time and effort required to create insightful analytics.

Notable features of AI/BI Dashboards include:

- **AI-Assisted BI Development:** By helping analysts optimize SQL, fix code errors and build visualizations using natural language, AI accelerates dashboard development, enabling faster decision-making and reducing manual work

- **Customizable Visualizations:** A wide range of visualization types allows users to present data in ways that are most relevant to their business context

- **Interactive Filters and Query-Based Parameters:** These features enable more granular analysis, giving users the ability to adjust their reports in real time to reflect specific business questions

- **Real-Time Data Exploration:** Dashboards operate on live data, ensuring that the insights reflect current business conditions

AI/BI Genie: The Future of Self-Service Analytics

The introduction of AI/BI Genie — a conversational AI analyst — is one of the most transformative elements of Databricks AI/BI. Designed to bridge the gap between nontechnical business users and complex data systems, Genie allows users to ask questions about their enterprise data in natural language and receive immediate, contextually relevant answers. This generative AI capability goes beyond basic query translation by learning from user interactions, continuously refining its understanding of business data and context.

Genie's ability to understand the full data estate allows it to return highly accurate results, even when dealing with complex queries. This enhanced understanding is referred to as *data intelligence*. Unlike many AI models that might generate incorrect or ambiguous answers, Genie also proactively seeks clarification from the user in instances of uncertainty, learning from feedback to ensure the integrity of the insights it delivers.

The Role of Generative AI in Enabling Self-Service Analytics

The power of generative AI within Databricks AI/BI lies in its capacity to extend self-service analytics to nontechnical users. Traditional BI tools often require users to possess technical skills, particularly in querying data through SQL or navigating predefined reports. Genie's natural language interface changes this dynamic by enabling business users to directly engage with data.

However, Genie's effectiveness isn't just due to its conversational interface — its awareness of the entire data platform is what truly sets it apart. Because Genie learns from the organization's data ecosystem, including usage patterns and business semantics, it can deliver insights with a high degree of accuracy, ensuring that even complex data queries are handled with precision.

databricks

Governance and Security: A Unified Approach

Databricks AI/BI inherits the governance and security capabilities of the Databricks Platform, particularly through Unity Catalog. This native integration ensures that all analytics are governed by consistent access controls and security policies. By operating within a unified platform, organizations benefit from:

- Consistent governance, as all data, from raw datasets to final visualizations, is subject to the same access policies

- Simplified compliance, as the platform supports auditing, lineage tracking and role-based access controls

- A reduction in risks associated with data silos, as all analytics activities occur within the same governed environment

By combining AI-driven analytics with robust governance, Databricks AI/BI provides users with flexibility and administrators with confidence in maintaining data security and integrity.

Best Practices for Maximizing the Potential of AI/BI

Organizations looking to maximize the potential of Databricks AI/BI should focus on establishing a culture of data literacy and self-service analytics. This can be achieved through:

- Training business users to effectively interact with Genie, ensuring they can independently query data and understand the results

- Leveraging AI/BI Dashboards to monitor critical business metrics, with regular reviews to adjust dashboards to reflect evolving business needs

- Utilizing Unity Catalog to enforce data governance across all departments, ensuring compliance and data security while providing seamless access to insights

For data practitioners wanting to create effective Genie spaces in Databricks AI/BI, our advice is to start small and iteratively expand based on feedback, using focused tables and well-annotated columns. Involve domain experts, define the audience and purpose, and thoroughly test with sample questions to ensure Genie interprets domain-specific language. Conduct user testing for continuous refinement, and address issues such as ambiguous jargon, table selection and performance bottlenecks. For details on troubleshooting and additional strategies, please see the full guidelines on Genie best practices.

databricks

Real-World Applications of AI/BI

Databricks AI/BI is already driving innovation across several industries, including:

- **Retail and Consumer Goods:** AI/BI enables retailers to optimize inventory levels by predicting consumer behavior and demand fluctuations in real time
- **Healthcare and Life Sciences:** Healthcare providers are leveraging AI/BI's capabilities to analyze patient data, enhancing diagnostics and improving treatment outcomes
- **Financial Services:** In the financial sector, AI/BI is being used to assess risk more effectively and detect fraud, thanks to its ability to analyze vast datasets with high accuracy

Examples like these highlight how AI/BI's deep integration with Databricks, combined with its AI-first approach, is transforming industries by enabling faster, more accurate decision-making.

The Future of AI-Driven Business Intelligence

As data continues to grow in volume and complexity, AI-first BI tools like Databricks AI/BI are poised to lead the future of analytics. By combining AI-powered insights with seamless integration into the data platform, Databricks AI/BI ensures that organizations are equipped to handle the demands of modern data environments. The evolution of self-service analytics, driven by generative AI models, will continue to push the boundaries of what business users can achieve, empowering them to uncover actionable insights without the need for technical expertise.

LEARN MORE

- Introducing AI/BI: Intelligent Analytics for Real-World Data
- Onboarding Your New AI/BI Genie
- Building Confidence in Your Genie Space With Benchmarks and Ask for Review
- How to Share AI/BI Dashboards With Everyone in Your Organization
- How to Embed AI/BI Dashboards Into Your Websites and Applications

databricks

Data Warehousing Best Practices on the Lakehouse

databricks

SECTION 4.1

Data Warehousing Modeling Techniques and Their Implementation on the Lakehouse

Using Data Vaults and Star Schemas on the Lakehouse

by Soham Bhatt and Deepak Sekar

The lakehouse is a new data platform paradigm that combines the best features of data lakes and data warehouses. It is designed as a large-scale enterprise-level data platform that can house many use cases and data products. It can serve as a single unified enterprise data repository for all of your:

- data domains,
- real-time streaming use cases,
- data marts,
- disparate data warehouses,
- data science feature stores and data science sandboxes, and
- departmental self-service analytics sandboxes.

Given the variety of the use cases, different data organizing principles and modeling techniques may apply to different projects on a lakehouse. Technically, the lakehouse architecture can support many different data modeling styles. In this article, we aim to explain the implementation of the Bronze/Silver/Gold data organizing principles of the lakehouse and how different data modeling techniques fit in each layer.
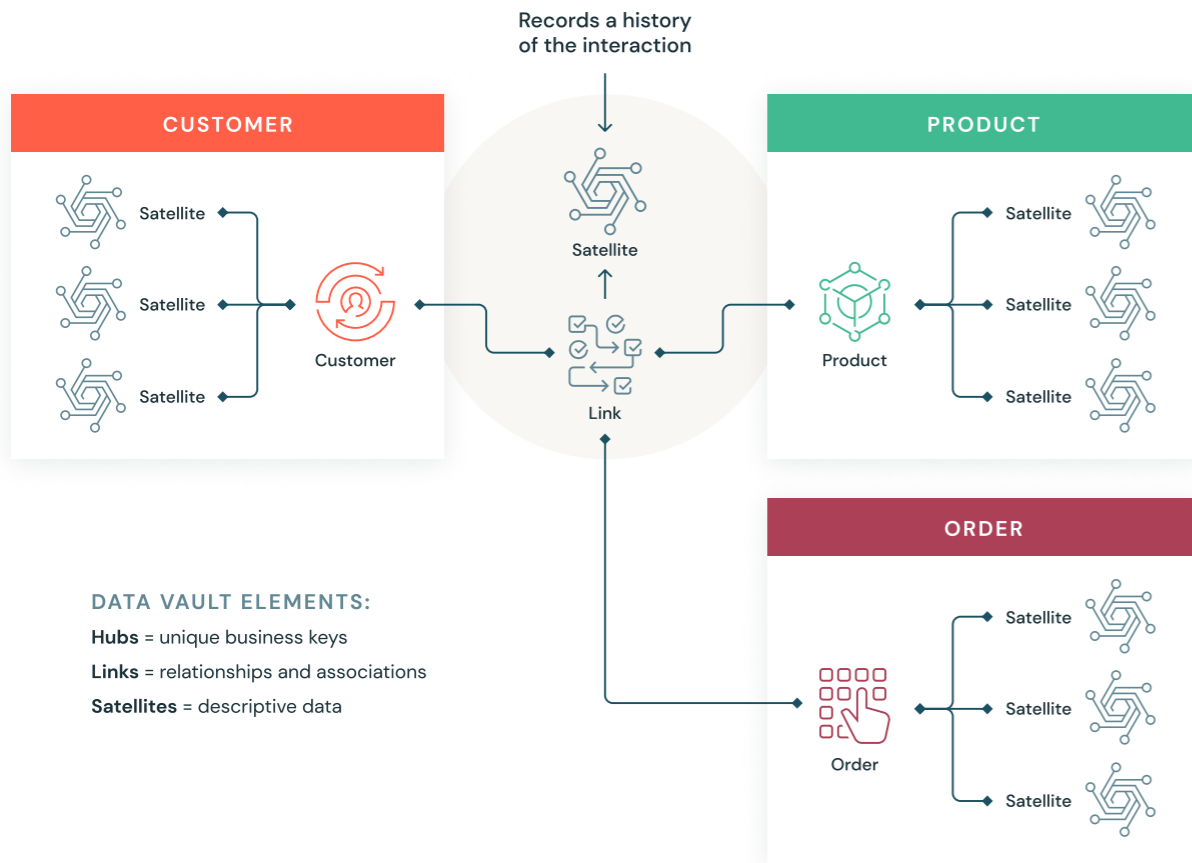
What is a Data Vault?

A Data Vault is a more recent data modeling design pattern used to build data warehouses for enterprise-scale analytics compared to Kimball and Inmon methods.

Data Vaults organize data into three different types: hubs, links, and satellites. Hubs represent core business entities, links represent relationships between hubs, and satellites store attributes about hubs or links.

Data Vault focuses on agile data warehouse development where scalability, data integration/ETL and development speed are important. Most customers have a landing zone, Vault zone and a data mart zone which correspond to the Databricks organizational paradigms of Bronze, Silver and Gold layers. The Data Vault modeling style of hub, link and satellite tables typically fits well in the Silver layer of the lakehouse architecture.

Learn more about Data Vault modeling at Data Vault Alliance.

databricks

Data vault modeling

Records a history
of the interaction

DATA VAULT ELEMENTS:

Hubs = unique business keys

**Links** = relationships and associations

Satellites = descriptive data

A diagram showing how Data Vault modeling works, with hubs, links, and satellites connecting to one another.

What is Dimensional Modeling?

Dimensional modeling is a bottom-up approach to designing data warehouses in order to optimize them for analytics. Dimensional models are used to denormalize business data into dimensions (like time and product) and facts (like transactions in amounts and quantities), and different subject areas are connected via conformed dimensions to navigate to different fact tables.

The most common form of dimensional modeling is the star schema. A star schema is a multi-dimensional data model used to organize data so that it is easy to understand and analyze, and very easy and intuitive to run reports on. Kimball-style star schemas or dimensional models are pretty much the gold standard for the presentation layer in data warehouses and data marts, and even semantic and reporting layers. The star schema design is optimized for querying large data sets.
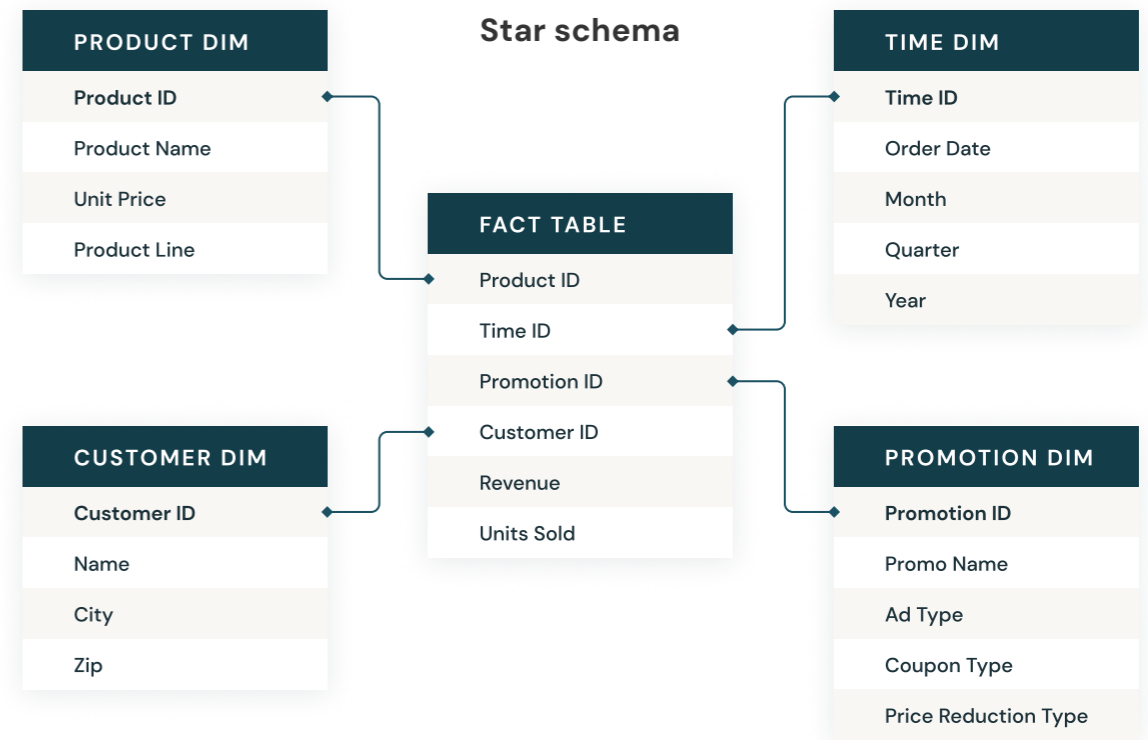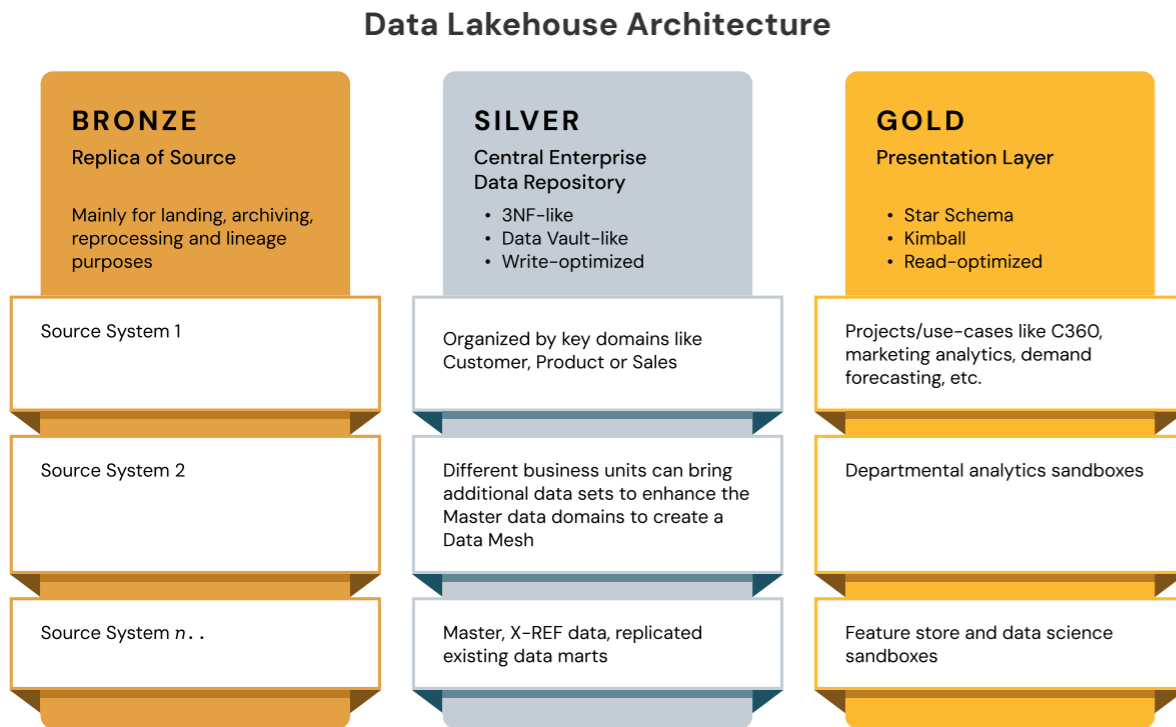
Star schema

A star schema example

Both normalized Data Vault (write-optimized) and denormalized dimensional models (read-optimized) data modeling styles have a place in the Databricks Data Intelligence Platform. The Data Vault's hubs and satellites in the Silver layer are used to load the dimensions in the star schema, and the Data Vault's link tables become the key driving tables to load the fact tables in the dimension model. Learn more about dimensional modeling from the Kimball Group.

Data organization principles in each layer of the lakehouse

A modern lakehouse is an all-encompassing enterprise-level data platform. It is highly scalable and performant for all kinds of different use cases such as ETL, BI, data science and streaming that may require different data modeling approaches. Let's see how a typical lakehouse is organized:

Data Lakehouse Architecture

BRONZE
Replica of Source

Mainly for landing, archiving, reprocessing and lineage purposes

- Source System 1
- Source System 2
- Source System n . .

SILVER
Central Enterprise Data Repository

- 3NF-like
- Data Vault-like
- Write-optimized

- Organized by key domains like Customer, Product or Sales
- Different business units can bring additional data sets to enhance the Master data domains to create a Data Mesh
- Master, X-REF data, replicated existing data marts

GOLD
Presentation Layer

- Star Schema
- Kimball
- Read-optimized

- Projects/use-cases like C360, marketing analytics, demand forecasting, etc.
- Departmental analytics sandboxes
- Feature store and data science sandboxes

A diagram showing characteristics of the Bronze, Silver and Gold layers of the data lakehouse architecture.

Bronze Layer — the Landing Zone

The Bronze layer is where we land all the data from source systems. The table structures in this layer correspond to the source system table structures "as-is," aside from optional metadata columns that can be added to capture the load date/time, process ID, etc. The focus in this layer is on change data capture (CDC), and the ability to provide an historical archive of source data (cold storage), data lineage, auditability, and reprocessing if needed — without rereading the data from the source system.

In most cases, it's a good idea to keep the data in the Bronze layer in Delta format, so that subsequent reads from the Bronze layer for ETL are performant — and so that you can do updates in Bronze to write CDC changes. Sometimes, when data arrives in JSON or XML formats, we do see customers landing it in the original source data format and then stage it by changing it to Delta format. So sometimes, we see customers manifest the logical Bronze layer into a physical landing and staging zone.

Storing raw data in the original source data format in a landing zone also helps with consistency wherein you ingest data via ingestion tools that don't support Delta as a native sink or where source systems dump data onto object stores directly. This pattern also aligns well with the autoloader ingestion framework wherein sources land the data in landing zone for raw files and then Databricks AutoLoader converts the data to Staging layer in Delta format.

databricks

Silver Layer — the Enterprise Central Repository

In the Silver layer of the lakehouse architecture, the data from the Bronze layer is matched, merged, conformed and cleaned ("just-enough") so that the Silver layer can provide an "enterprise view" of all its key business entities, concepts and transactions. This is akin to an Enterprise Operational Data Store (ODS) or a Central Repository or Data domains of a Data Mesh (e.g. master customers, products, non-duplicated transactions and cross-reference tables). This enterprise view brings the data from different sources together, and enables self-service analytics for ad-hoc reporting, advanced analytics and ML. It also serves as a source for departmental analysts, data engineers and data scientists to further create data projects and analysis to answer business problems via enterprise and departmental data projects in the Gold layer.

In the lakehouse data engineering paradigm, typically the (Extract-Load-Transform) ELT methodology is followed vs. traditional Extract-Transform-Load(ETL). ELT approach means only minimal or "just-enough" transformations and data cleansing rules are applied while loading the Silver layer. All the "enterprise level" rules are applied in the Silver layer vs. project-specific transformational rules, which are applied in the Gold layer. Speed and agility to ingest and deliver the data in the lakehouse is prioritized here.

From a data modeling perspective, the Silver layer has more 3rd-Normal Form like data models. Data Vault-like write-performant data architectures and data models can be used in this layer. If using a Data Vault methodology, both the raw Data Vault and Business Vault will fit in the logical Silver layer of the lake — and the Point-In-Time (PIT) presentation views or materialized views will be presented in the Gold layer.

Gold Layer — the Presentation Layer

In the Gold layer, multiple data marts or warehouses can be built as per dimensional modeling/Kimball methodology. As discussed earlier, the Gold layer is for reporting and uses more denormalized and read-optimized data models with fewer joins compared to the Silver layer. Sometimes tables in the Gold layer can be completely denormalized, typically if the data scientists want it that way to feed their algorithms for feature engineering.

ETL and data quality rules that are "project-specific" are applied when transforming data from the Silver layer to Gold layer. Final presentation layers such as data warehouses, data marts or data products like customer analytics, product/quality analytics, inventory analytics, customer segmentation, product recommendations, marketing/sales analytics, etc., are delivered in this layer. Kimball style star-schema based data models or Inmon style Data marts fit in this Gold layer of the lakehouse. Data Science Laboratories and Departmental Sandboxes for self-service analytics also belong in the Gold layer.

databricks

## The Lakehouse Data Organization Paradigm

To summarize, data is curated as it moves through the different layers of a data lakehouse architecture.

- The Bronze layer uses the data models of source systems. If data is landed in raw formats, it is converted to DeltaLake format within this layer.

- The Silver layer for the first time brings the data from different sources together and conforms it to create an Enterprise view of the data — typically using a more normalized, write-optimized data models that are typically 3rd-Normal Form-like or Data Vault-like.

- The Gold layer is the presentation layer with more denormalized or flattened data models than the Silver layer, typically using Kimball-style dimensional models or star schemas. The Gold layer also houses departmental and data science sandboxes to enable self-service analytics and data science across the enterprise. Providing these sandboxes and their own separate compute clusters prevents the Business teams from creating their own copies of data outside of the Lakehouse.
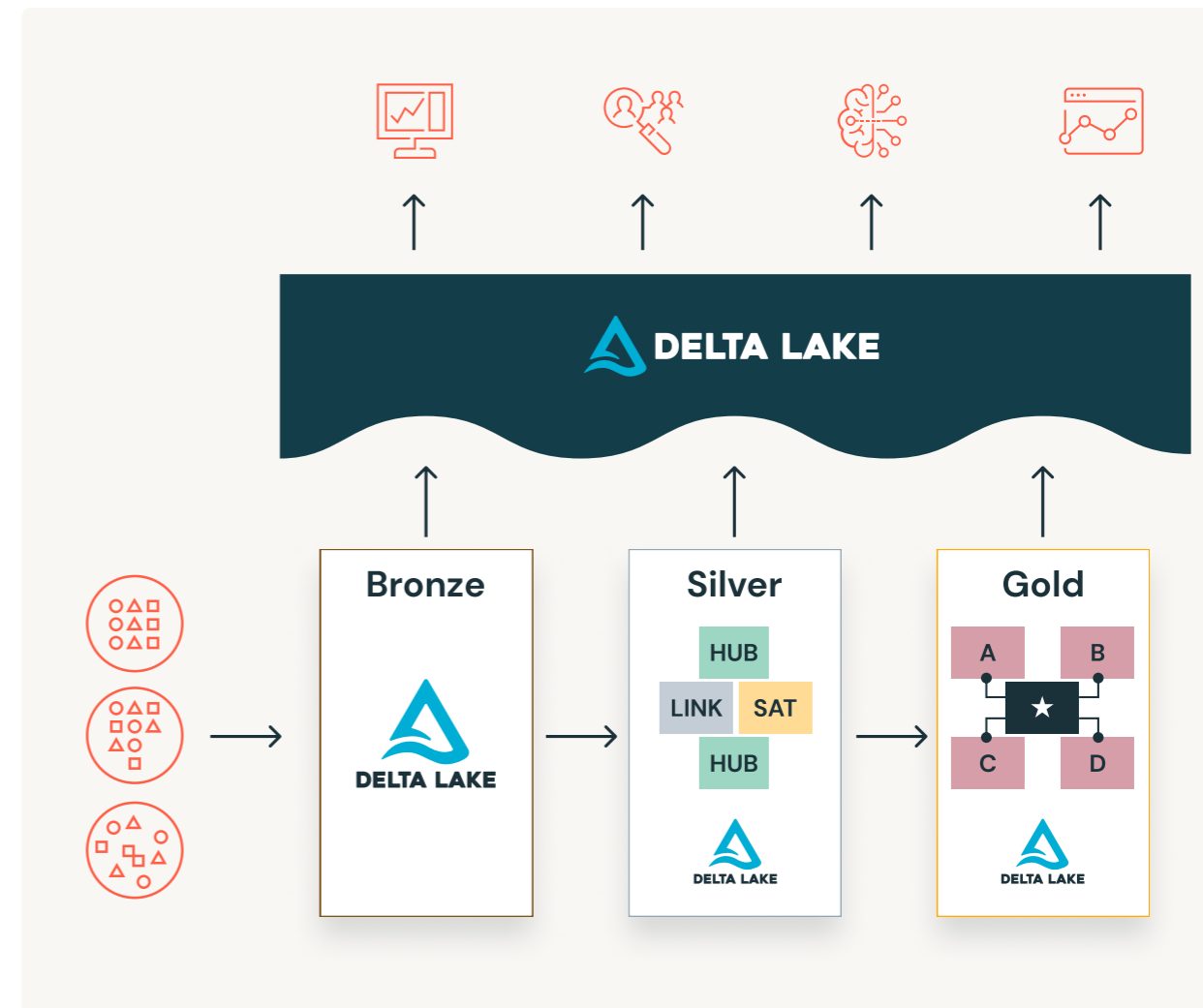
This lakehouse data organization approach is meant to break data silos, bring teams together, and empower them to do ETL, streaming, and BI and AI on one platform with proper governance. Central data teams should be the enablers of innovation in the organization, speeding up the onboarding of new self-service users, as well as the development of many data projects in parallel — rather than the data modeling process becoming the bottleneck. The Databricks Unity Catalog provides search and discovery, governance and lineage on the lakehouse to ensure good data governance cadence.

Build your Data Vaults and star schema data warehouses with Databricks SQL today. →
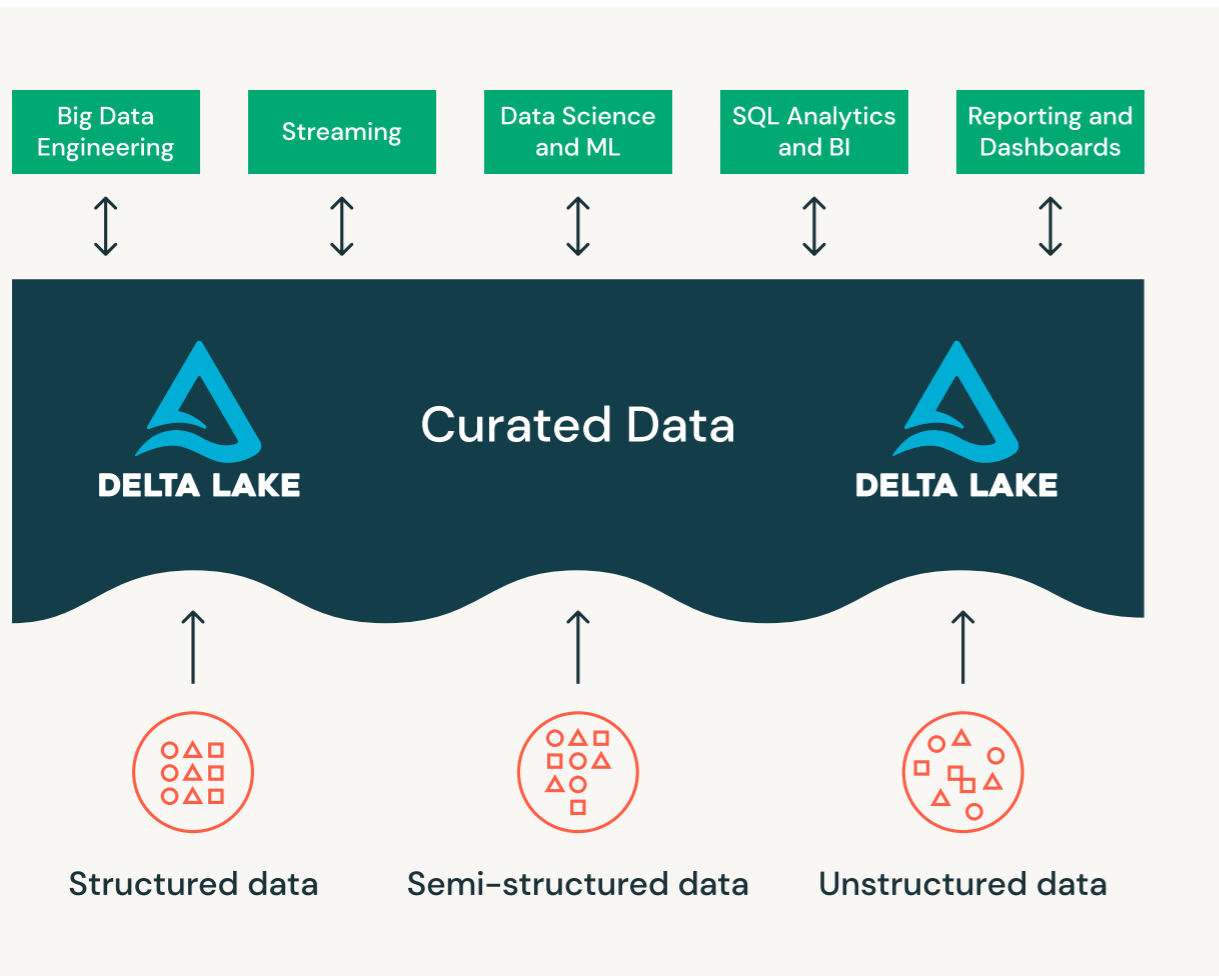
How data is curated as it moves through the various lakehouse architecture layers.

LEARN MORE

→ Five Simple Steps for Implementing a Star Schema in Databricks With Delta Lake

→ Best practices to implement a Data Vault model in Databricks Lakehouse

→ Dimensional Modeling Best Practice & Implementation on Modern Lakehouse

→ Identity Columns to Generate Surrogate Keys Are Now Available in a Lakehouse Near You!

→ Load an EDW Dimensional Model in Real Time With Databricks Lakehouse

SECTION 4.2

Dimensional Modeling Best Practice and Implementation on a Modern Lakehouse Architecture

by Leo Mao, Abhishek Dey, Justin Breese and Soham Bhatt

A large number of our customers are migrating their legacy data warehouses to Databricks Lakehouse as it enables them to modernize not only their Data Warehouse but they also instantly get access to a mature Streaming and Advanced Analytics platform. Lakehouse can do it all as it is one platform for all your streaming, ETL, BI, and AI needs — and it helps your business and Data teams collaborate on one platform.

As we help customers in the field, we find that many are looking for best practices around proper data modeling and physical data model implementations in Databricks.

In this article, we aim to dive deeper into the best practice of dimensional modeling on the Databricks Data Intelligence Platform and provide a live example of a physical data model implementation using our table creation and DDL best practices.

Here are the high-level topics we will cover in this blog:

- The Importance of Data Modeling
- Common Data Modeling Techniques
- Data Warehouse Modeling DDL Implementation
- Best Practice and Recommendation for Data Modeling on the Lakehouse

The importance of Data Modeling for Data Warehouse

Data Models are front and center of building a Data Warehouse. Typically the process starts with defending the Semantic Business Information Model, then a Logical data Model, and finally a Physical Data Model (PDM). It all starts with a proper Systems Analysis and Design phase where a Business Information model and process flows are created first and key business entities, attributes and their interactions are captured as per the business processes within the organization. The Logical Data Model is then created depicting how the entities are related to each other and this is a Technology agnostic model. Finally a PDM is created based on the underlying technology platform to ensure that the writes and reads can be performed efficiently. As we all know, for Data Warehousing, Analytics-friendly modeling styles like Star-schema and Data Vault are quite popular.

databricks

Best practices for creating a Physical Data Model in Databricks

Based on the defined business problem, the aim of the data model design is to represent the data in an easy way for reusability, flexibility, and scalability. Here is a typical star-schema data model that shows a Sales fact table that holds each transaction and various dimension tables such as customers, products, stores, date etc. by which you slice-and-dice the data. The dimensions can be joined to the fact table to answer specific business questions such as what are the most popular products for a given month or which stores are best performing ones for the quarter. Let's see how to get it implemented in Databricks.
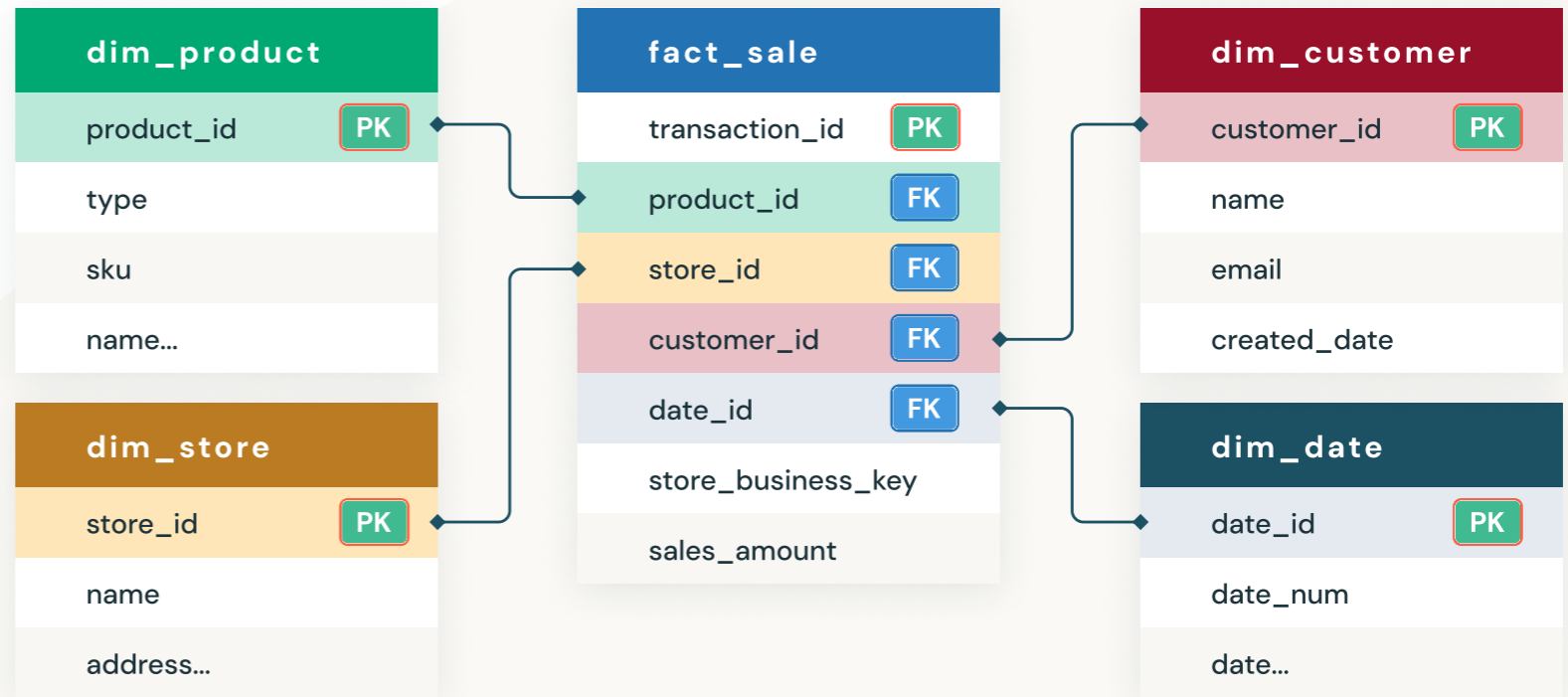
Dimensional model on the lakehouse architecture

Note each dimension table has **__START_AT** and **__END_AT** columns to support SCD Type 2, which are not displayed here because of limited space.

databricks

Data Warehouse Modeling DDL Implementation on Databricks

In the following sections, we would demonstrate the below using our examples.

- Creating 3-level Catalog, Database and Table
- Primary Key, Foreign Key definitions
- Identity columns for Surrogate keys
- Column constraints for Data Quality
- Index, optimize and analyze
- Advanced techniques

1. Unity Catalog – 3 level namespace

Unity Catalog is a Databricks Governance layer which lets Databricks admins and data stewards manage users and their access to data centrally across all of the workspaces in a Databricks account using one Metastore. Users in different workspaces can share access to the same data, depending on privileges granted centrally in Unity Catalog. Unity Catalog has 3 level Namespace (catalog.schema(database).table) that organizes your data. Learn more about Unity Catalog here.
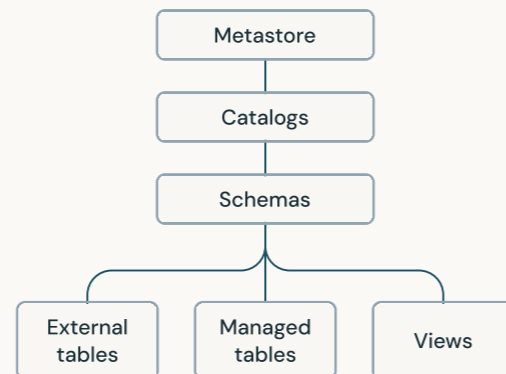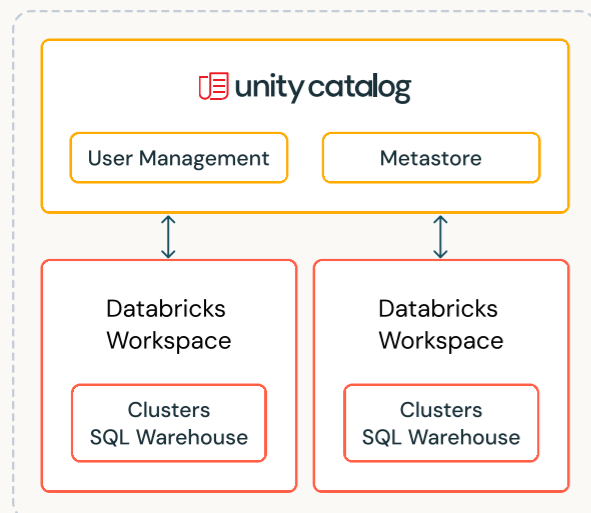
Here is how to set up the catalog and schema before we create tables within the database. For our example, we create a catalog **US_Stores** and a schema (database) **Sales_DW** as below, and use them for the later part of the section.

```
1   CREATE CATALOG IF NOT EXISTS US_Stores;
2   USE CATALOG US_Stores;
3   CREATE SCHEMA IF NOT EXISTS Sales_DW;
4   USE SCHEMA Sales_DW;
```

Setting up the Catalog and Database

Here is an example on querying the **fact_sales** table with a 3 level namespace.

```
SELECT *
FROM US_Stores.Sales_DW.fact_sales
```

▸ (2) Spark Jobs

Table   Data Profile

	transaction_id	date_id	customer_id	product_id	store_id	store_business_key	sales_amount
1	10001	20211001	1	1	1	PER01	50
2	10004	20211003	2	1	2	BNE02	60
3	10005	20211003	3	2	1	PER01	79
4	10002	20211002	2	1	2	BNE02	79
5	10003	20211002	1	2	2	BNE02	79

Showing all 5 rows.

Example of querying table with catalog.database.tablename

databricks

2. Primary Key, Foreign Key definitions

Primary and Foreign Key definitions are very important when creating a data model. Having the ability to support the PK/FK definition makes defining the data model super easy in Databricks. It also helps analysts quickly figure out the join relationships in Databricks SQL Warehouse so that they can effectively write queries. Like most other Massively Parallel Processing (MPP), EDW, and Cloud Data Warehouses, the PK/FK constraints are informational only. Databricks does not support enforcement of the PK/FK relationship, but gives the ability to define it to make the designing of Semantic Data Model easy.

Here is an example of creating the **dim_store** table with **store_id** as an Identity Column and it's also defined as a Primary Key at the same time.

```
1   -- Store dimension
2   CREATE OR REPLACE TABLE dim_store(
3     store_id BIGINT GENERATED ALWAYS AS IDENTITY PRIMARY KEY,
4     business_key STRING,
5     name STRING,
6     email STRING,
7     city STRING,
8     address STRING,
9     phone_number STRING,
10    created_date TIMESTAMP,
11    updated_date TIMESTAMP,
12    start_at TIMESTAMP,
13    end_at TIMESTAMP
14  );
```

DDL Implementation for creating store dimension with Primary Key Definitions

After the table is created, we can see that the primary key (store_id) is created as a constraint in the table definition below.

Describe Dim Store table information

DESC TABLE EXTENDED US_Stores.Sales_DW.dim_store

Table   Data Profile

	col_name	data_type
20	Owner	leo.mao@databricks.com
21	Is_managed_location	true
22	Table Properties	[delta.minReaderVersion=1,delta.minWriterVersion=6]
23		
24	# Constraints	
25	dim_store_pk	PRIMARY KEY (`store_id`)

Showing all 25 rows.

Primary Key store_id shows as table constraint

Here is an example of creating the **fact_sales** table with **transaction_id** as a Primary Key, as well as foreign keys that are referencing the dimension tables.

```
1   -- Fact Sales
2   CREATE OR REPLACE TABLE fact_sales(
3     transaction_id BIGINT PRIMARY KEY,
4     date_id BIGINT NOT NULL CONSTRAINT dim_date_fk FOREIGN KEY REFERENCES dim_date,
5     customer_id BIGINT NOT NULL CONSTRAINT dim_customer_fk FOREIGN KEY REFERENCES
6   dim_customer,
7     product_id BIGINT NOT NULL CONSTRAINT dim_product_fk FOREIGN KEY REFERENCES
8   dim_product,
9     store_id BIGINT NOT NULL CONSTRAINT dim_store_fk FOREIGN KEY REFERENCES dim_
10  store,
11    store_business_key STRING,
12    sales_amount DOUBLE
13  );
```

DDL Implementation for creating sales fact with Foreign Key definitions

databricks

After the fact table is created, we could see that the primary key **(transaction_id)** and foreign keys are created as constraints in the table definition below.

Describe Fact Sales Table Info

```
DESC TABLE EXTENDED US_Stores.Sales_DW.fact_sales
```

Table   Data Profile

	col_name	data_type
20	# Constraints	
21	dim_product_fk	FOREIGN KEY (`product_id`) REFERENCES `us_stores`.`sales_dw`.`dim_product` (`product_id`)
22	dim_date_fk	FOREIGN KEY (`date_id`) REFERENCES `us_stores`.`sales_dw`.`dim_date` (`date_id`)
23	dim_customer_fk	FOREIGN KEY (`customer_id`) REFERENCES `us_stores`.`sales_dw`.`dim_customer` (`customer_id`)
24	dim_store_fk	FOREIGN KEY (`store_id`) REFERENCES `us_stores`.`sales_dw`.`dim_store` (`store_id`)
25	fact_sales_pk	PRIMARY KEY (`transaction_id`)

Showing all 25 rows.

Fact table definition with primary key and foreign keys referencing dimensions

3. Identity columns for Surrogate keys

An identity column is a column in a database that automatically generates a unique ID number for each new row of data. These are commonly used to create surrogate keys in the data warehouses. Surrogate keys are system-generated, meaningless keys so that we don't have to rely on various Natural Primary Keys and concatenations on several fields to identify the uniqueness of the row. Typically these surrogate keys are used as Primary and Foreign keys in data warehouses. Details on Identity columns are discussed in this blog. Below is an example of creating an identity column customer_id, with automatically assigned values starting with 1 and increment by 1.

```
1   -- Customer dimension
2   CREATE OR REPLACE TABLE dim_customer(
3     customer_id BIGINT GENERATED ALWAYS AS IDENTITY (START WITH 1 INCREMENT BY 1)
4   PRIMARY KEY,
5     name STRING,
6     email STRING,
7     address STRING,
8     created_date TIMESTAMP,
9     updated_date TIMESTAMP,
10    start_at TIMESTAMP,
11    end_at TIMESTAMP
12  );
```

DDL Implementation for creating customer dimension with identity column

databricks

4. Column constraints for Data Quality

In addition to Primary and Foreign key informational constraints, Databricks also supports column-level Data Quality Check constraints which are enforced to ensure the quality and integrity of data added to a table. The constraints are automatically verified. Good examples of these are NOT NULL constraints and column value constraints. Unlike the other Cloud Data Warehouse, Databricks went further to provide column value check constraints, which are very useful to ensure the data quality of a given column. As we could see below, the **valid_ sales_amount** check constraint will verify that all existing rows satisfy the constraint (i.e. **sales amount** > 0) before adding it to the table. More information can be found here.

Here are examples to add constraints for dim_store and fact_sales respectively to make sure store_id and sales_amount have valid values.

```
1  -- Add constraint to dim_store to make sure column store_id is between 1 and 9998
2  ALTER TABLE US_Stores.Sales_DW.dim_store ADD CONSTRAINT valid_store_id CHECK
3  (store_id > 0 and store_id < 9999);
4
5  -- Add constraint to fact_sales to make sure column sales_amount has a valid value
6  ALTER TABLE US_Stores.Sales_DW.fact_sales ADD CONSTRAINT valid_sales_amount CHECK
7  (sales_amount > 0);
```

Add column constraint to existing tables to ensure data quality

5. Index, Optimize, and Analyze

Traditional Databases have b-tree and bitmap indexes, Databricks has much advanced form of indexing - multi-dimensional Z-order clustered indexing and we also support Bloom filter indexing. First of all, the Delta file format uses Parquet file format, which is a columnar compressed file format so it's already very efficient in column pruning and on top of it using z-order indexing gives you the ability to sift through petabyte scale data in seconds. Both Z-order and Bloom filter indexing dramatically reduce the amount of data that needs to be scanned in order to answer highly selective queries against large Delta tables, which typically translates into orders-of-magnitude runtime improvements and cost savings. Use Z-order on your Primary Keys and foreign keys that are used for the most frequent joins. And use additional Bloom filter indexing as needed.

```
1  -- Optimise fact_sales table by customer_id and product_id for better query and
2  join performance
3  OPTIMIZE US_Stores.Sales_DW.fact_sales
4  ZORDER BY (customer_id, product_id);
```

Optimize fact_sales on customer_id and product_id for better performance

```
1  -- Create a bloomfilter index to enable data skipping on store_business_key
2  CREATE BLOOMFILTER INDEX
3  ON TABLE US_Stores.Sales_DW.fact_sales
4  FOR COLUMNS(store_business_key)
```

Create a Bloomfilter Index to enable data skipping on a given column

databricks

And just like any other Data warehouse, you can ANALYZE TABLE to update statistics to ensure the Query optimizer has the best statistics to create the best query plan.

```sql
1   -- collect stats for all columns for better performance
2   ANALYZE TABLE US_Stores.Sales_DW.fact_sales COMPUTE STATISTICS FOR ALL COLUMNS;
```

Collect stats for all the columns for better query execution plan

6. Advanced Techniques

While Databricks support advanced techniques like Table Partitioning, please use these feature sparingly, only when you have many Terabytes of compressed data – because most of the time our OPTIMIZE and Z-ORDER indexes will give you the best file and data pruning which makes partitioning a table by date or month almost a bad practice. It is however a good practice to make sure that your table DDLs are set for auto optimization and auto compaction. These will ensure your frequently written data in small files are compacted into bigger columnar compressed formats of Delta.

Are you looking to leverage a visual data modeling tool? Our partner erwin Data Modeler by Quest can be used to reverse engineer, create and implement Star-schema, Data Vaults, and any Industry Data Models in Databricks with just a few clicks.

Databricks Notebook Example

With the Databricks Platform, one can easily design and implement various data models with ease. To see all of the above examples in a complete workflow, please look at this example.

Please also check out our related blog:
Five Simple Steps for Implementing a Star Schema in Databricks With Delta Lake →

databricks

SECTION 4.3

Loading a Data Warehouse Data Model in Real Time With the Databricks Data Intelligence Platform

Dimensional modeling implementation on the modern lakehouse using Delta Live Tables

by Leo Mao, Soham Bhatt and Abhishek Dey

Dimensional modeling is one of the most popular data modeling techniques for building a modern data warehouse. It allows customers to quickly develop facts and dimensions based on business needs for an enterprise. When helping customers in the field, we found many are looking for best practices and implementation reference architecture from Databricks.

In this article, we aim to dive deeper into the best practice of dimensional modeling on the lakehouse architecture and provide a live example to load an EDW dimensional model in real-time using Delta Live Tables.

Here are the high-level steps we will cover in this blog:

- Define a business problem
- Design a dimensional model
- Best practices and recommendations for dimensional modeling
- Implementing a dimensional model on a lakehouse architecture
- Conclusion

databricks

Define a business problem

Dimensional modeling is business-oriented; it always starts with a business problem. Before building a dimensional model, we need to understand the business problem to solve, as it indicates how the data asset will be presented and consumed by end users. We need to design the data model to support more accessible and faster queries.

The Business Matrix is a fundamental concept in Dimensional Modeling, below is an example of the business matrix, where the columns are shared dimensions and rows represent business processes. The defined business problem determines the grain of the fact data and required dimensions. The key idea here is that we could incrementally build additional data assets with ease based on the Business Matrix and its shared or conformed dimensions.

Business Processes	Date	Customer	Product	Vendor	Promotion	Reseller	Sales Territory	Employee	Account	Organization
Internet Sales	✓	✓	✓	✓	✓		✓			
Reseller Sales	✓		✓		✓	✓	✓	✓		
General Ledger	✓								✓	✓
Sales Plan	✓		✓				✓			
Inventory	✓		✓	✓					✓	
Customer Surveys	✓	✓								
Customer Service Calls	✓	✓	✓					✓		

A Business Matrix with Shared Dimensions and Business Processes

Here we assume that the business sponsor would like to team to build a report to give insights on:

- What are the top selling products so they can understand product popularity

- What are the best performing stores to learn good store practices

Design a dimensional model

Based on the defined business problem, the data model design aims to represent the data efficiently for reusability, flexibility and scalability. Here is the high-level data model that could solve the business questions above.
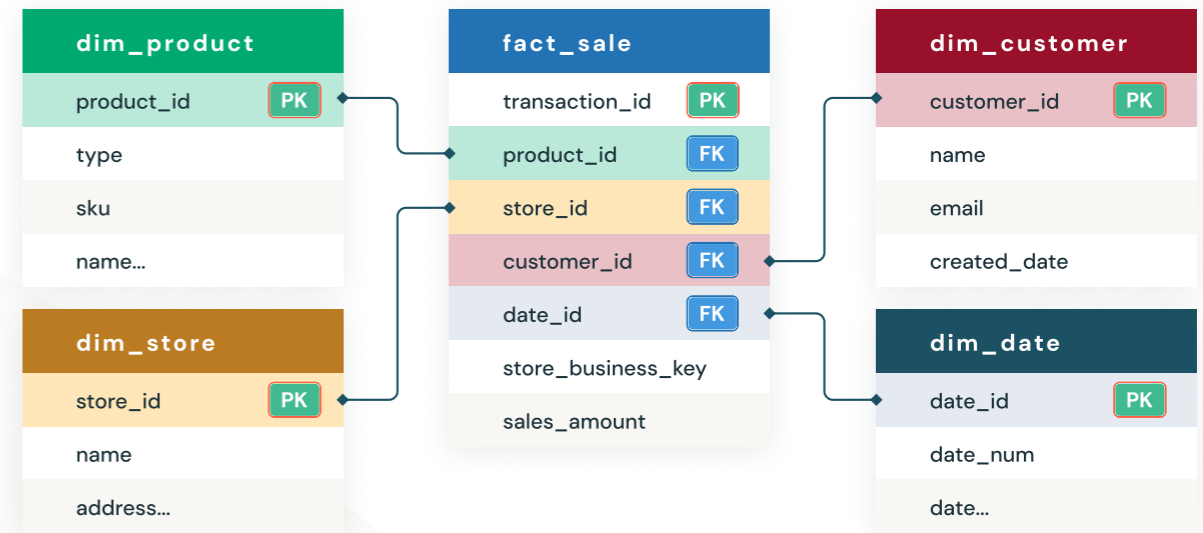
Dimensional model on the lakehouse architecture

Note each dimension table has __START_AT and __END_AT columns to support SCD Type 2, which are not displayed here because of limited space.

The design should be easy to understand and efficient with different query patterns on the data. From the model, we designed the sales fact table to answer our business questions; as you can see, other than the foreign keys (FKs) to the dimensions, it only contains the numeric metrics used to measure the business, e.g. sales_amount.

We also designed dimension tables such as Product, Store, Customer, Date that provide contextual information on the fact data. Dimension tables are typically joined with fact tables to answer specific business questions, such as the most popular products for a given month, which stores are the best-performing ones for the quarter, etc.

databricks

Best practices and recommendations for dimensional modeling

With the Databricks Data Intelligence Platform, one can easily design and implement dimensional models, and simply build the facts and dimensions for the given subject area.

Below are some of the best practices recommended while implementing a dimensional model:

- One should denormalize the dimension tables. Instead of the third normal form or snowflake type of model, dimension tables typically are highly denormalized with flattened many-to-one relationships within a single dimension table.

- Use conformed dimension tables when attributes in different dimension tables have the same column names and domain contents. This advantage is that data from different fact tables can be combined in a single report using conformed dimension attributes associated with each fact table.

- A usual trend in dimension tables is around tracking changes to dimensions over time to support as-is or as-was reporting. You can easily apply the following basic techniques for handling dimensions based on different requirements.
  - The type 1 technique overwrites the dimension attribute's initial value.

  - With the type 2 technique, the most common SCD technique, you use it for accurate change tracking over time.

  This can be easily achieved out of the box with Delta Live Tables implementation.
  - One can easily perform SCD type 1 or SCD type 2 using Delta Live Tables using APPLY CHANGES INTO

- Primary + Foreign Key **Constraints** allow end users like yourselves to understand relationships between tables.

- **Usage of IDENTITY Columns** automatically generates unique integer values when new rows are added. Identity columns are a form of surrogate keys. Refer to the blog link for more details.

- **Enforced CHECK Constraints** to never worry about data quality or data correctness issues sneaking up on you.

databricks

Implementing a dimensional model on a lakehouse architecture

Now, let us look at an example of Delta Live Tables based dimensional modeling implementation:

The example code below shows us how to create a dimension table (dim_store) using SCD Type 2, where change data is captured from the source system.

```
1   -- create the gold table
2   CREATE INCREMENTAL LIVE TABLE dim_store
3   TBLPROPERTIES ("quality" = "gold")
4   COMMENT "Slowly Changing Dimension Type 2 for store dimension in the gold layer";
5
6   -- store all changes as SCD2
7   APPLY CHANGES INTO live.dim_store
8   FROM STREAM(live.silver_store)
9     KEYS (store_id)
10    SEQUENCE BY updated_date
11    COLUMNS * EXCEPT (_rescued_data, input_file_name)
12    STORED AS SCD TYPE 2;
```

The example code below shows us how to create a fact table (fact_sale), with the constraint of valid_product_id we are able to ensure all fact records that are loaded have a valid product associated with it.

```
1   -- create the fact table for sales in gold layer
2   CREATE STREAMING LIVE TABLE fact_sale (
3     CONSTRAINT valid_store_business_key EXPECT (store_business_key IS NOT NULL) ON
4   VIOLATION DROP ROW,
5     CONSTRAINT valid_product_id EXPECT (product_id IS NOT NULL) ON VIOLATION DROP
6   ROW
7   )
8   TBLPROPERTIES ("quality" = "gold", "ignoreChanges" = "true")
9   COMMENT "sales fact table in the gold layer" AS
10    SELECT
11      sale.transaction_id,
12      date.date_id,
13      customer.customer_id,
14      product.product_id AS product_id,
15      store.store_id,
16      store.business_key AS store_business_key,
17      sales_amount
18    FROM STREAM(live.silver_sale) sale
19    INNER JOIN live.dim_date date
20    ON to_date(sale.transaction_date, 'M/d/yy') = to_date(date.date, 'M/d/yyyy')
21    -- only join with the active customers
22    INNER JOIN (SELECT * FROM live.dim_customer WHERE __END_AT IS NULL) customer
23    ON sale.customer_id = customer.customer_id
24    -- only join with the active products
25    INNER JOIN (SELECT * FROM live.dim_product WHERE __END_AT IS NULL) product
26    ON sale.product = product.SKU
27    -- only join with the active stores
28    INNER JOIN (SELECT * FROM live.dim_store WHERE __END_AT IS NULL) store
29    ON sale.store = store.business_key
```


databricks

The Delta Live Tables pipeline example could be found here. Please refer to Delta Live Tables quickstart on how to create a Delta Live Tables pipeline. As seen below, DLT offers full visibility of the ETL pipeline and dependencies between different objects across Bronze, Silver and Gold layers following the lakehouse medallion architecture.
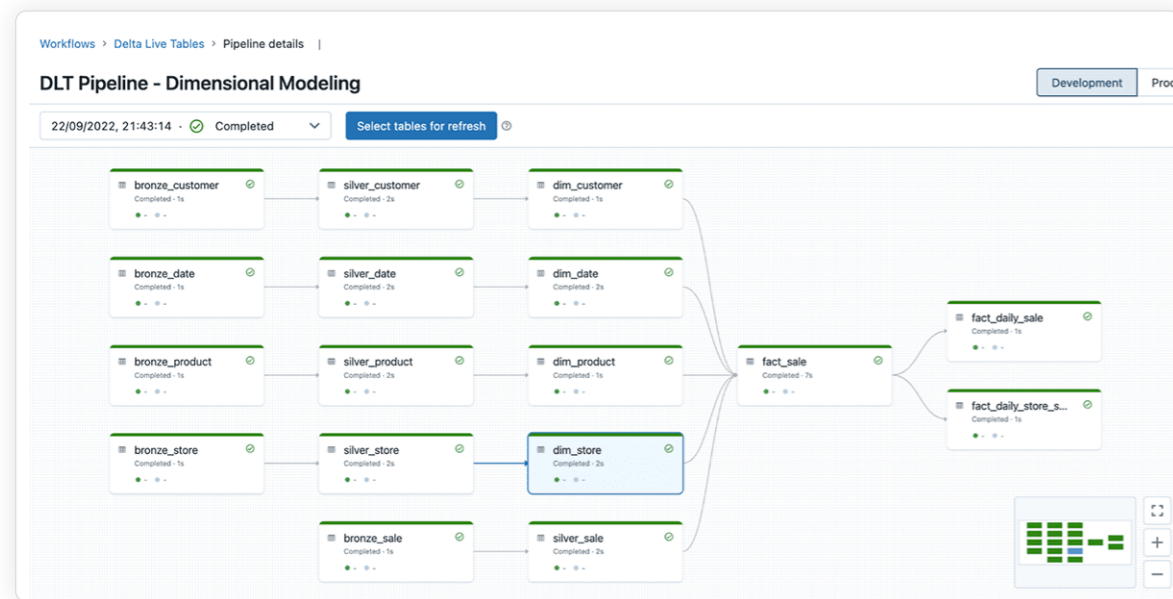
End to End DLT Pipeline

Here is an example of how the dimension table dim_store gets updated based on the incoming changes. Below, the Store Brisbane Airport was updated to Brisbane Airport V2, and with the out-of-box SCD Type 2 support, the original record ended on Jan 07 2022, and a new record was created which starts on the same day with an open end date (NULL) – which indicates the latest record for the Brisbane airport.

SCD Type 2 for Store Dimension

For more implementation details, please refer to here for the full notebook example.

Conclusion

In this blog, we learned about dimensional modeling concepts in detail, best practices, and how to implement them using Delta Live Tables.

Learn more about dimensional modeling at Kimball Technology.

databricks

SECTION 4.4

What's New With Databricks SQL?

AI-driven optimizations, lakehouse federation, and more for enterprise-grade BI

by Alex Lichen, Miranda Luna, Can Efeoglu and Cyrielle Simeone

At this year's Data + AI Summit, Databricks SQL continued to push the boundaries of what a data warehouse can be, leveraging AI across the entire product surface to extend our leadership in performance and efficiency, while still simplifying the experience and unlocking new opportunities for our customers. In parallel, we continue to deliver improvements to our core data warehousing capabilities to help you unify your data stack under the lakehouse architecture on the Databricks Data Intelligence Platform.

In this blog post, we are thrilled to share the highlights of what's new and coming next in Databricks SQL:

- AI-driven performance optimizations like Predictive I/O that deliver leading performance and cost-savings, without manual tuning required.

- New user experiences with AI Functions, SQL Warehouses in Notebooks, new Dashboards, and Python User Defined Functions.

- Rich external data source support with Lakehouse Federation.

- New ways to access your data with the SQL Statement Execution API.

- Simple and efficient data processing with Streaming Tables, Materialized Views and Workflows integration.

- Intelligent assistance powered by DatabricksIQ, our knowledge engine.

- Enhanced administration tools with Databricks SQL System Tables and Live Query Profile.

- Additional features for our partner integrations with Fivetran, dbt labs and PowerBI

The AI-optimized warehouse: Ready for all your workloads — no tuning required

We believe that the best data warehouse is a lakehouse; therefore, we continue to extend our leadership in ETL workloads and harnessing the power of AI. Databricks SQL now also delivers industry-leading performance for your EDA and BI workloads, while improving cost savings — with no manual tuning.

Best TCO & Performance across use cases

ETL	EDA	BI
Up to **9x** Less Cost **2–3x** Faster	Up to **2x** Less Cost **5–10%** Faster	Up to **10x** Faster **10x** Queries/minute
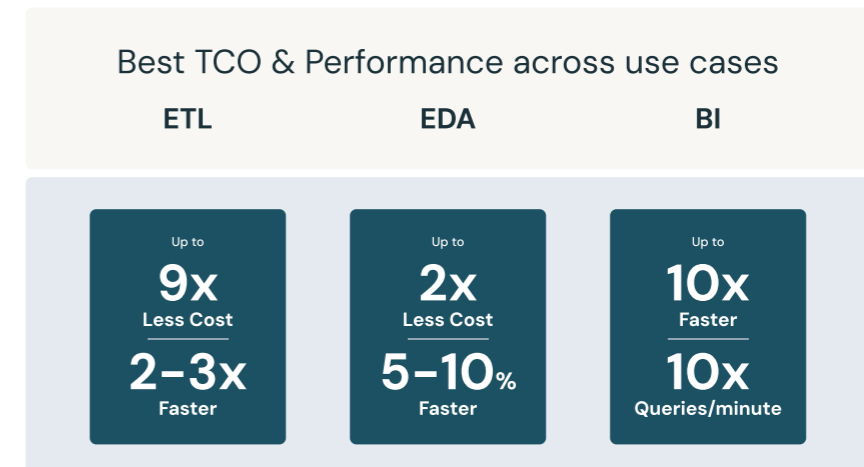
Say goodbye to manually creating indexes. With Predictive I/O for reads (GA) and updates (Public Preview), Databricks SQL now analyzes historical read and write patterns to intelligently build indexes and optimize workloads. Early customers have benefited from a remarkable 35x improvement in point lookup efficiency, impressive performance boosts of 2-6x for MERGE operations and 2-10x for DELETE operations.

databricks

With Predictive Optimizations (Public Preview), Databricks will seamlessly optimize file sizes and clustering by running OPTIMIZE, VACUUM, ANALYZE and CLUSTERING commands for you. With this feature, Anker Innovations benefited from a 2.2x boost to query performance while delivering 50% savings on storage costs.

> **Databricks' Predictive Optimizations intelligently optimized our Unity Catalog storage, which saved us 50% in annual storage costs while speeding up our queries by >2x. It learned to prioritize our largest and most-accessed tables. And, it did all of this automatically, saving our team valuable time.**
>
> — ANKER INNOVATIONS

Tired of managing different warehouses for smaller and larger workloads or fine tuning scaling parameters? Intelligent Workload Management is a suite of features that keeps queries fast while keeping cost low. By analyzing real time patterns, Intelligent Workload Management ensures that your workloads have the optimal amount of compute to execute incoming SQL statements without disrupting already running queries.

With AI-powered optimizations, Databricks SQL provides industry leading TCO and performance for any kind of workload, without any manual tuning needed. To learn more about available optimization previews, watch Reynold Xin's keynote and Databricks SQL Serverless Under the Hood: How We Use ML to Get the Best Price/Performance from the Data+AI Summit.

Unlock siloed data with Lakehouse Federation

Today's organizations face challenges in discovering, governing and querying siloed data sources across fragmented systems. With Lakehouse Federation, data teams can use Databricks SQL to discover, query and manage data in external platforms including MySQL, PostgreSQL, Amazon Redshift, Snowflake, Azure SQL Database, Azure Synapse, Google's BigQuery (coming soon) and more.

Furthermore, Lakehouse Federation seamlessly integrates with advanced features of Unity Catalog when accessing external data sources from within Databricks. Enforce row and column level security to restrict access to sensitive information. Leverage data lineage to trace the origins of your data and ensure data quality and compliance. To organize and manage data assets, easily tag federated catalog assets for simple data discovery.

Finally, to accelerate complicated transformations or cross-joins on federated sources, Lakehouse Federation supports Materialized Views for better query latencies.

For more details, watch our dedicated session Lakehouse Federation: Access and Governance of External Data Sources from Unity Catalog from the Data+AI Summit.

databricks

Develop on the lakehouse architecture with the SQL Statement Execution API

The SQL Statement Execution API enables access to your Databricks SQL warehouse over a REST API to query and retrieve results. With HTTP frameworks available for almost all programming languages, you can easily connect to a diverse array of applications and platforms directly to a Databricks SQL Warehouse.

The Databricks SQL Statement Execution API is available with the Databricks Premium and Enterprise tiers. To learn more, watch our session, follow our tutorial (AWS | Azure), read the documentation (AWS | Azure), or check our repository of code samples.

Streamline your data processing with Streaming Tables, Materialized Views, and DB SQL in Workflows

With Streaming Tables, Materialized Views, and DB SQL in Workflows, any SQL user can now apply data engineering best practices to process data. Efficiently ingest, transform, orchestrate, and analyze data with just a few lines of SQL.

Streaming Tables are the ideal way to bring data into "Bronze" tables. With a single SQL statement, scalably ingest data from various sources such as cloud storage (S3, ADLS, GCS), message buses (EventHub, Kafka, Kinesis), and more. This ingestion occurs incrementally, enabling low-latency and cost-effective pipelines, without the need for managing complex infrastructure.

```
1   CREATE STREAMING TABLE web_clicks
2   AS
3   SELECT *
4   FROM STREAM
5       read_files('s3://mybucket')
```

Materialized Views reduce cost and improve query latency by pre-computing slow queries and frequently used computations, and are incrementally refreshed to improve overall latency. In a data engineering context, they are used for transforming data. But they are also valuable for analyst teams in a data warehousing context because they can be used to (1) speed up end-user queries and BI dashboards, and (2) securely share data. In just four lines of code, any user can create a materialized view for performant data processing.

```
1   CREATE MATERIALIZED VIEW customer_orders
2   AS
3   SELECT
4       customers.name,
5       sum(orders.amount),
6       orders.orderdate
7   FROM orders
8       LEFT JOIN customers ON
9           orders.custkey = customers.c_custkey
10  GROUP BY
11      name,
12      orderdate;
```

Need orchestration with DB SQL? Workflows now allows you to schedule SQL queries, dashboards and alerts. Easily manage complex dependencies between tasks and monitor past job executions with the intuitive Workflows UI or via API.

Streaming Tables and Materialized Views are now in public preview. To learn more, read our dedicated blog post. To enroll in the public preview for both, enroll in this form. Workflows in DB SQL is now generally available, and you can learn more by reading the documentation (AWS | Azure).

databricks

Databricks Assistant: Write better and faster SQL with natural language

Databricks Assistant is a context-aware AI assistant embedded inside Databricks Notebooks and the SQL Editor. Databricks Assistant can take a natural language question and suggest a SQL query to answer that question. When trying to understand a complex query, users can ask the Assistant to explain it using natural language, enabling anyone to understand the logic behind query results.
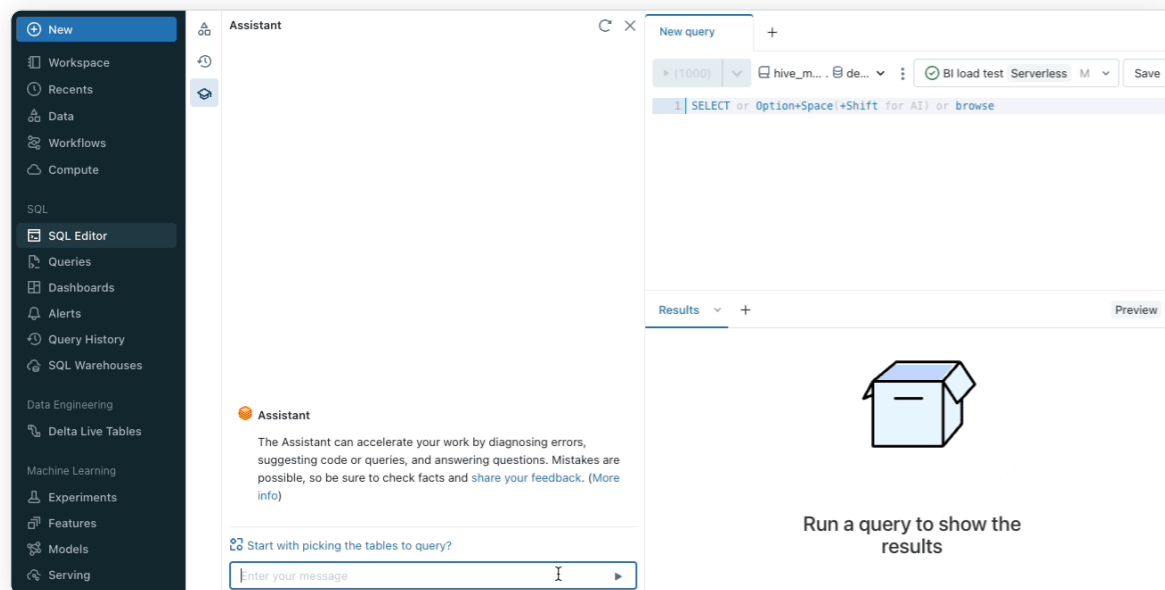
Databricks Assistant uses a number of signals to provide more accurate, relevant results. It uses context from code cells, libraries, popular tables, Unity Catalog schemas and tags to map natural language questions into queries and code.

In the future, we will be adding integration with DatabricksIQ to provide even more context for your requests.

Manage your data warehouse with confidence

Administrators and IT teams need the tools to understand data warehouse usage. With System Tables, Live Query Profile, and Statement Timeouts, admins can monitor and fix problems when they occur, ensuring that your data warehouse runs efficiently.

Gain deeper visibility and insights into your SQL environment with System Tables. System Tables are Databricks-provided tables that contain information about past statement executions, costs, lineage, and more. Explore metadata and usage metrics to answer questions like "What statements were run and by whom?", "How and when did my warehouses scale?" and "What was I billed for?". Since System Tables are integrated within Databricks, you have access to native capabilities such as SQL alerts and SQL dashboards to automate the monitoring and alerting process.

As of today, there are three System Tables currently in public preview: Audit Logs, Billable Usage System Table, and Lineage Sytem Table (AWS | Azure). Additional system tables for warehouse events and statement history are coming soon.

databricks

For example, to compute the monthly DBUs used per SKU, you can query the Billiable Usage System Tables.

```
1   SELECT sku_name, usage_date, sum(usage_quantity) as `DBUs`
2       FROM system.billing.usage
3   WHERE
4       month(usage_date) = month(NOW())
5       AND year(usage_date) = year(NOW())
6   GROUP BY sku_name, usage_date
```

With Live Query Profile, users gain real-time insights into query performance to help optimize workloads on the fly. Visualize query execution plans and assess live query task executions to fix common SQL mistakes like exploding joins or full table scans. Live Query Profile allows you to ensure that running queries on your data warehouse are optimized and running efficiently. Learn more by reading the documentation (AWS | Azure).

Looking for automated controls? Statement Timeouts allow you to set a custom workspace or query level timeout. If a query's execution time exceeds the timeout threshold, the query will be automatically halted. Learn more by reading the documentation (AWS | Azure)

Compelling new experiences in DBSQL

Over the past year, we've been hard at work to add new, cutting-edge experiences to Databricks SQL. We're excited to announce new features that put the power of AI in SQL users hands such as, enabling SQL warehouses throughout the entire Databricks platform; introducing a new generation of SQL dashboards; and bringing the power of Python into Databricks SQL.

Democratize unstructured data analysis with AI Functions

With AI Functions, DB SQL is bringing the power of AI into the SQL warehouse. Effortlessly harness the potential of unstructured data by performing tasks such as sentiment analysis, text classification, summarization, translation and more. Data analysts can apply AI models via self-service, while data engineers can independently build AI-enabled pipelines.

Using AI Functions is quite simple. For example, consider a scenario where a user wants to classify the sentiment of some articles into Frustrated, Happy, Neutral, or Satisfied.

```
1   -- create a udf for sentiment classification
2   CREATE FUNCTION classify_sentiment(text STRING)
3     RETURNS STRING
4     RETURN ai_query(
5       'Dolly', -- the name of the model serving endpoint
6       named_struct(
7         'prompt',
8         CONCAT('Classify the following text into one of four categories [Frustrated,
9   Happy, Neutral, Satisfied]:\n',
10          text),
11        'temperature', 0.5),
12      'returnType', 'STRING');
```

```
1   -- use the udf
2   SELECT classify_sentiment(text) AS sentiment
3   FROM reviews;
```

AI Functions are now in Public Preview. To sign up for the Preview, fill out the form here. To learn more, you can also read our detailed blog post or review the documentation (AWS | Azure).

databricks

Bring the power of SQL warehouses to notebooks

Databricks SQL warehouses are now public preview in notebooks, combining the flexibility of notebooks with the performance and TCO of Databricks SQL Serverless and Pro warehouses. To enable SQL warehouses in notebooks, simply select an available SQL warehouse from the notebooks compute dropdown.

Connecting serverless SQL warehouses from Databricks notebooks

Find and share insights with a new generation of dashboards

Discover a revamped dashboarding experience directly on the lakehouse architecture. Users can simply select a desired dataset and build stunning visualizations with a SQL–optional experience. Say goodbye to managing separate queries and dashboard objects – an all-in-one content model

simplifies the permissions and management process. Finally, publish a dashboard to your entire organization, so that any authenticated user in your identity provider can access the dashboard via a secure web link, even if they don't have Databricks access.

New Databricks SQL Dashboards are currently in Private Preview. Contact your account team to learn more.

Leverage the flexibility of Python in SQL

Bring the flexibility of Python into Databricks SQL with Python user–defined functions (UDFs). Integrate machine learning models or apply custom redaction logic for data processing and analysis by calling custom Python functions directly from your SQL query. UDFs are reusable functions, enabling you to apply consistent processing to your data pipelines and analysis.

For instance, to redact email and phone numbers from a file, consider the following CREATE FUNCTION statement.

```
1   CREATE FUNCTION redact(a STRING)
2   RETURNS STRING
3   LANGUAGE PYTHON
4   AS $$
5   import json
6   keys = ["email", "phone"]
7   obj = json.loads(a)
8   for k in obj:
9     if k in keys:
10      obj[k] = "REDACTED"
11  return json.dumps(obj)
12  $$;
```

Learn more about enrolling in the private preview here.

Integrations with your data ecosystem

At Data+AI Summit, Databricks SQL announced new integrations for a seamless experience with your tools of choice.

Databricks + Fivetran

We're thrilled to announce the general availability of Fivetran access in Partner Connect for all users including non-admins with sufficient privileges to a catalog. This innovation makes it 10x easier for all users to ingest data into Databricks using Fivetran. This is a huge win for all Databricks customers as they can now bring data into the lakehouse architecture from hundreds of connectors Fivetran offers, like Salesforce and PostgreSQL. Fivetran now fully supports serverless warehouses as well!

> **Learn more by reading the blog post here.**

Databricks + dbt Labs

Simplify real-time analytics engineering on the lakehouse architecture with Databricks and dbt Labs. The combination of dbt's highly popular analytics engineering framework with the Databricks Platform provides powerful capabilities:

- dbt + Streaming Tables: Streaming ingestion from any source is now built-in to dbt projects. Using SQL, analytics engineers can define and ingest cloud/streaming data directly within their dbt pipelines.

- dbt + Materialized Views: Building efficient pipelines becomes easier with dbt, leveraging Databricks' powerful incremental refresh capabilities. Users can use dbt to build and run pipelines backed by MVs, reducing infrastructure costs with efficient, incremental computation.

> **To learn more, read the detailed blog post.**

databricks

Databricks + PowerBI: Publish to PowerBI Workspaces

Publish datasets from your Databricks workspace to PowerBI Online workspace with a few clicks! No more managing odbc/jdbc connections — simply select the dataset you want to publish. Simply select the datasets or schema you want to publish and select your PBI workspace! This makes it easier for BI admins and report creators to support PowerBI workspaces without also having to use Power BI Desktop.
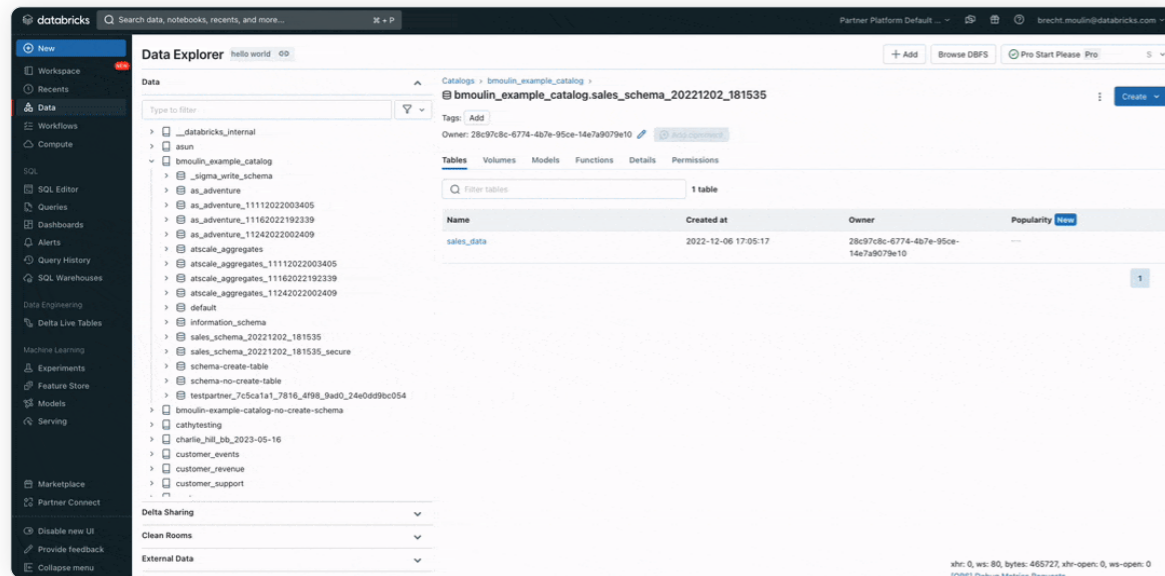
Getting Started with Databricks SQL

Follow the guide (AWS | Azure | GCP) on how to setup a SQL warehouse to get started with Databricks SQL today! Databricks SQL Serverless is currently available with a 20%+ promotional discount, visit our pricing page to learn more.

You can also watch **Databricks SQL: Why the Best Serverless Data Warehouse is a Lakehouse** and **What's New in Databricks SQL — With Live Demos** for a complete overview.

SECTION 4.5

Distributed Data Governance and Isolated Environments With Unity Catalog

by Max Nienu, Zeashan Pappa, Paul Roome and Sachin Thakur

Effective data governance is essential for any organization that relies on data, analytics and AI for its operations. In many organizations, there is a growing recognition of the value proposition of centralized data governance. However, even with the best intentions, implementing centralized governance can be challenging without the proper organizational processes and resources. The role of Chief Data Officer (CDO) is still emerging in many organizations, leaving questions about who will define and execute data governance policies across the organization.

As a result, the responsibility for defining and executing data governance policies across the organization is often not centralized, leading to policy variations or governing bodies across lines of business, sub-units, and other divisions within an organization. For simplicity, we can call this pattern distributed governance, where there is a general agreement on the distinctions between these governing units but not necessarily a central data governance function.

In this blog, we'll explore implementing a distributed governance model using Databricks Unity Catalog, which provides a unified governance solution for data, analytics, and AI in the lakehouse.

Evolution of Data Governance in Databricks

Before the introduction of Unity Catalog, the concept of a workspace was monolithic, with each workspace having its own metastore, user management, and Table ACL store. This led to intrinsic data and governance isolation boundaries between workspaces and duplication of effort to address consistency across them.

To handle this, some customers resorted to running pipelines or code to synchronize their metastores and ACLs, while others set up their own self-managed metastores to use across workspaces. However, these solutions added more overhead and maintenance costs forcing upfront architecture decisions on how to partition data across the organization, creating data silos.

Data Governance with Unity Catalog

To overcome these limitations, Databricks developed Unity Catalog, which aims to make it easy to implement data governance while maximizing the ability to collaborate on and share data. The first step in achieving this was implementing a common namespace that permits access to any data within an organization.

This approach may seem like a challenge to the distributed governance pattern mentioned earlier but Unity Catalog offers new isolation mechanisms within the namespace that organizations have traditionally addressed using multiple Hive metastores. These isolation mechanisms enable groups to operate independently with minimal or no interaction and also allow them to achieve isolation in other scenarios, such as production vs development environments.

Hive Metastore versus Unity Catalog in Databricks

With Hive, a metastore was a service boundary, meaning that having different metastores meant different hosted underlying Hive services and different underlying databases. Unity Catalog is a platform service within the Databricks Data Intelligence Platform, so there are no service boundaries to consider.

Unity Catalog provides a common namespace that allows you to govern and audit your data in one place.

When using Hive, it was common to use multiple metastores, each with its own namespace, to achieve isolation between development and production environments, or to allow for the separation of data between operating units.

In Unity Catalog, these requirements are solved through dynamic isolation mechanisms on namespaces that don't compromise the ability to share and collaborate on data and don't require hard one–way upfront architecture decisions.

Working across different teams and environments

When using a data platform, there is often a strong need to have isolation boundaries between environments like dev/prod and between business groups, teams, or operating units of your organization.

Let's begin by defining isolation boundaries in a data platform such as Databricks:

- Users should only gain access to data based on agreed access rules
- Data can be managed by designated people or teams
- Data should be physically separated in storage
- Data should only be accessed in designated environments

Users should only gain access to data based on agreed access rules

Organizations usually have strict requirements around data access based on some organizational/regulatory requirements which is fundamental to keeping data secure. Typical examples include employee salary information or credit card payment information.

Access to this type of information is typically tightly controlled and audited periodically. Unity Catalog provides organizations granular control over data assets within the catalog to meet these industry standards. With the controls, Unity Catalog provides users will only see and query the data they are entitled to see and query.

Data can be managed by designated people or teams

Unity Catalog gives you the ability to choose from centralized governance or distributed governance models.

In the centralized governance model, your governance administrators are owners of the metastore and can take ownership of any object and set ACLs and policy.

databricks

In a distributed governance model, you would consider a catalog or set of catalogs to be a data domain. The owner of that catalog can create and own all assets and manage governance within that domain. Therefore the owners of domains can operate independently of other owners in other domains.

We strongly recommend setting a group to be the owner or service principal for both of these options if management is done through tooling.
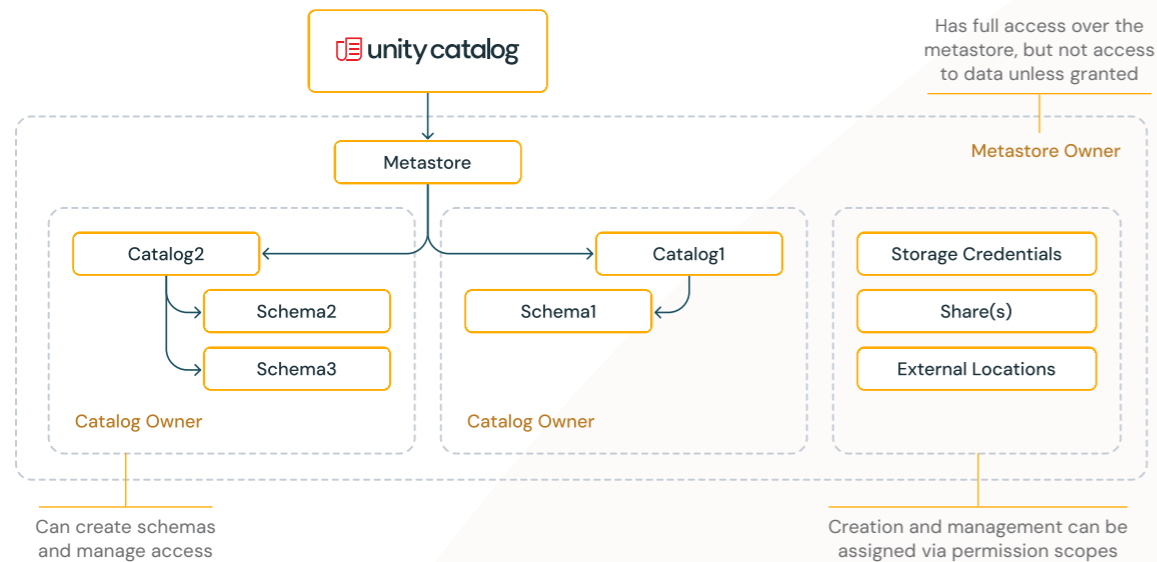
Data should be physically separated in storage

By default, when creating a UC metastore, the Databricks Account Admin provides a single cloud storage location and credential as the default location for managed tables.

Organizations that require physical isolation of data, for regulatory reasons, or for example across SDLC scopes, between business units, or even for cost allocation purposes, should consider managed data source features at the catalog and schema level.

Unity Catalog allows you to choose the defaults for how data is separated in storage. By default, all data is stored at the metastore. With feature support for managed data sources on catalogs and schemas, you can physically isolate data storage and access, helping your organization achieve their governance and data management requirements.

When creating managed tables, the data will then be stored using the schema location (if present) followed by the catalog location (if present), and will only use the metastore location if the prior two locations have not been set.

![databricks]
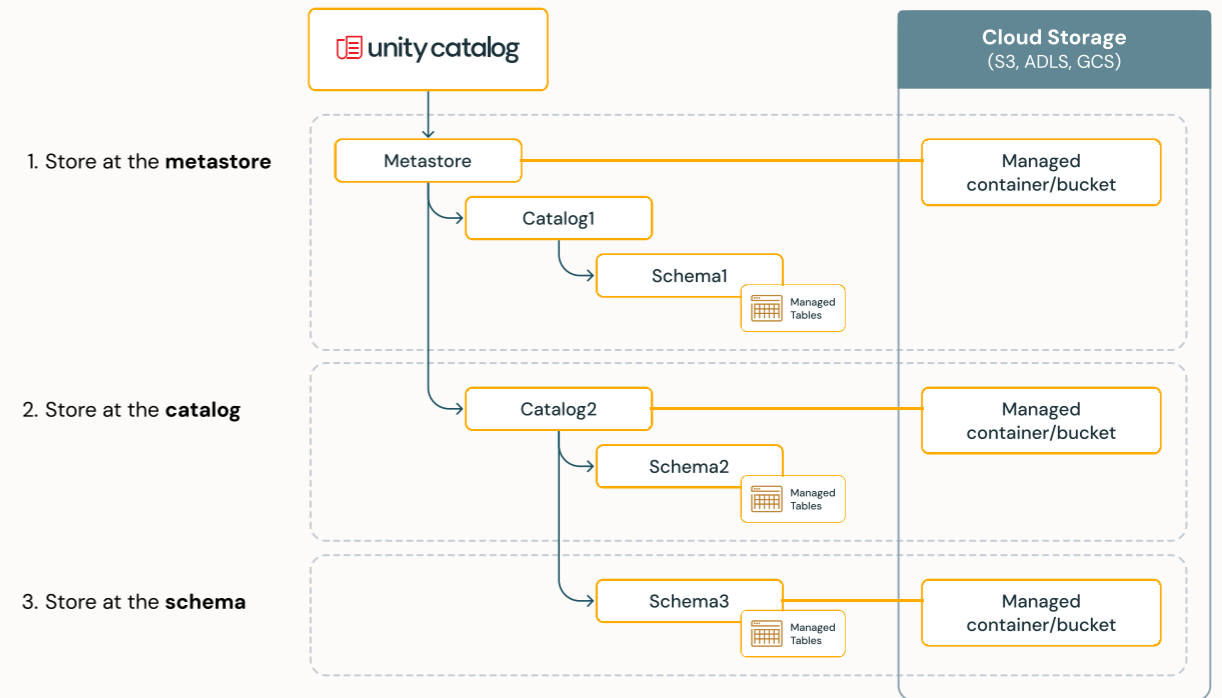
Data should only be accessed in designated environments, based on the purpose of that data

Oftentimes, organizational and compliance requirements maintain that you need to keep certain data accessible only in certain environments versus others. An example of this could be dev and production, or HIPAA or PII environments that contain PII data for analysis and have special access rules around who can access the data and the environments that allow access to that data. Sometimes requirements dictate that certain data sets or domains cannot be crossed or combined together.

In Databricks, we consider a workspace to be an environment. Unity Catalog has a feature that allows you to 'bind' catalogs to workspace. These environment–aware ACLs give you the ability to ensure that only certain catalogs are available within a workspace, regardless of a user's individual ACLs. This means that the metastore admin, or the catalog owner can define the workspaces that a data catalog can be accessed from. This can be controlled via our UI or via API/ terraform for easy integrations. We even recently published a blog on how to control Unity Catalog via terraform to help fit your specific governance model.

Access to data and availability of data can be isolated across workspaces and groups. Users can only access certain catalogs in certain environments.

Conclusion

With Unity Catalog at the center of your lakehouse architecture, you can achieve a flexible and scalable governance implementation without sacrificing your ability to manage and share data effectively. With Unity Catalog, you can overcome the limitations and constraints of your existing Hive metastore, enabling you to more easily isolate and collaborate on data according to your specific business needs. Follow the Unity Catalog guides (AWS, Azure) to get started. Download this free ebook on Data, analytics and AI governance to learn more about best practices to build an effective governance strategy for your data lakehouse.

databricks

SECTION 4.6

The Hitchhiker's Guide to Data Privilege Model and Access Control in Unity Catalog

Distill concepts within the Unity Catalog privilege model in a simple, digestible way, to support different access needs and patterns

by Som Natarajan and Vuong Nguyen

As the volume, velocity and variety of data grows, organizations are increasingly relying on staunch data governance practices to ensure their core business outcomes are adequately met. Unity Catalog is a fine-grained governance solution for data and AI powering the Databricks Data Intelligence Platform. It helps simplify the security and governance of your enterprise data assets by providing a centralized mechanism to administer and audit data access.

Taking a journey down memory lane, before Unity Catalog unified the permission model for files, tables and added support for all languages, customers were implementing fine-grained data access control on Databricks using the legacy workspace-level Table ACL (TACL), which were essentially restricted to certain cluster configurations and worked only for Python and SQL. Both Unity Catalog and TACL let you control access to securable objects like catalogs, schemas (databases), tables, views, but there are some nuances in how each access model works.

A good understanding of the object access model is essential for implementing data governance at scale using Unity Catalog. Even more so, if you have already implemented the Table ACL model and are looking to upgrade to Unity Catalog to take advantage of all the newest features, such as multi-language support, centralized access control and data lineage.

The Axioms of Unity Catalog access model

- Unity Catalog privileges are defined at metastore — Unity Catalog permissions always refer to account-level identities, while TACL permissions defined within the hive_metastore catalog always refer to the local identities in the workspace

- Privilege inheritance — Objects in Unity Catalog are hierarchical and privileges are inherited downward. The highest level object that privileges are inherited from is the catalog

- Object ownership is important — Privileges can only be granted by a metastore admin, the owner of an object, or the owner of the catalog or schema that contains the object. Only the owner of an object, or the owner of the catalog or schema that contains it can drop the object

- USE privileges for boundaries — USE CATALOG/SCHEMA is required to interact with objects within a catalog/schema. However, USE privilege does not allow one to browse the object metadata that is housed within the catalog/schema

- Permissions on derived objects are simplified — Unity Catalog only requires the owner of a view to have SELECT privilege, along with USE SCHEMA on the views' parent schema and USE CATALOG on the parent catalog. In contrast with TACL, a view's owner needs to be an owner of all referenced tables and views

databricks

Some more complex axioms

- Secure by default — only clusters with Unity-Catalog specific access modes (shared or single-user) can access Unity Catalog data. With TACL, all users have access to all data on non-shared clusters

- Limitation of single-user clusters — Single users clusters do not support dynamic views. Users must have SELECT on all referenced tables and views to read from a view

- No support for ANY FILE or ANONYMOUS FUNCTIONs: Unity Catalog does not support these permissions, as they could be used to circumvent access control restrictions by allowing an unprivileged user to run privileged code

Interesting patterns

There are many governance patterns that can be achieved using the Unity Catalog access model.

Example 1 – Consistent permissions across workspaces

Axiom 1 allows product team to define permissions for their data product within their own workspace, and having those reflected and enforced across all other workspaces, no matter where their consumers are coming from

Example 2 – Setting boundary for data sharing

Axiom 2 allows catalog/schema owners to set up default access rules for their data. For example the following commands enable the machine learning team to create tables within a schema and read each other's tables:
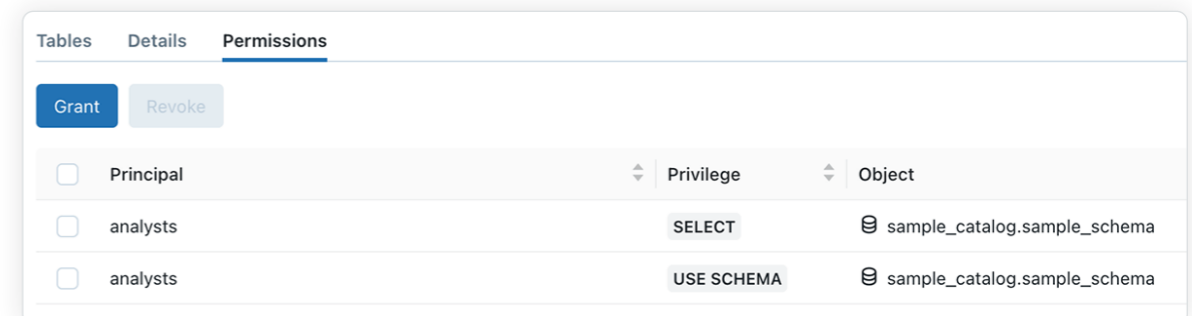
```
1   CREATE CATALOG ml;
2   CREATE SCHEMA ml.sandbox;
3   GRANT USE_CATALOG ON CATALOG ml TO ml_users;
4   GRANT USE_SCHEMA ON SCHEMA ml.sandbox TO ml_users;
5   GRANT CREATE TABLE ON SCHEMA ml.sandbox TO ml_users;
6   GRANT SELECT ON SCHEMA ml.sandbox TO ml_users;
```

More interestingly, axiom 4 now allows catalog/schema owners to limit how far individual schema and table owners can share data they produce. A table owner granting SELECT to another user does not allow that user read access to the table unless they also have been granted USE CATALOG privileges on its parent catalog as well as USE SCHEMA privileges on its parent schema.

In the below example, sample_catalog is owned by user A, user B created a sample_schema schema, and table 42. Even though USE SCHEMA and SELECT permission is granted to the analysts team, they still cannot query the table, due to permission boundary set by user A

Tables	Details	**Permissions**		
Grant	Revoke			
Principal			Privilege	Object
analysts			SELECT	sample_catalog.sample_schema
analysts			USE SCHEMA	sample_catalog.sample_schema

Permission page showing analysts group having SELECT and USE SCHEMA permission on sample_catalog.sample_schema

databricks
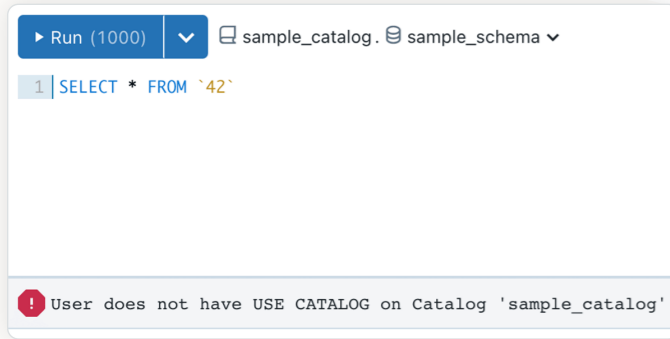
Query page with an error message

Example 3 – Easier sharing of business logic

Data consumers have a need to share their workings and transformation logic, and a reusable way of doing it is by creating and sharing views to other consumers.

Axiom 5 unlocks the ability for data consumers to do this seamlessly, without requiring manual back and forth with the table owners.
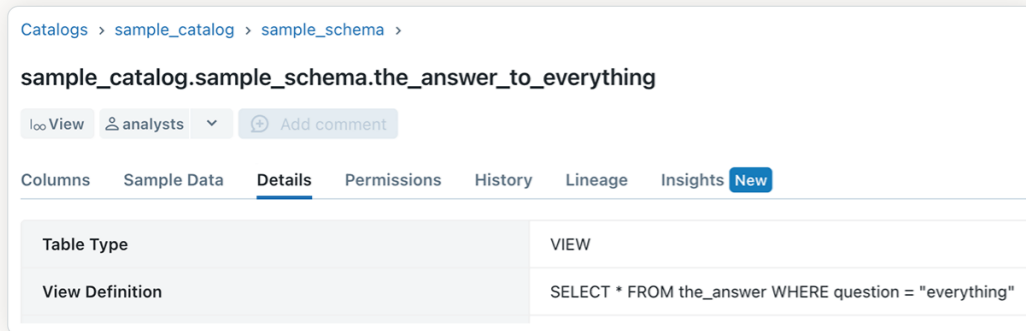
Definition of the view
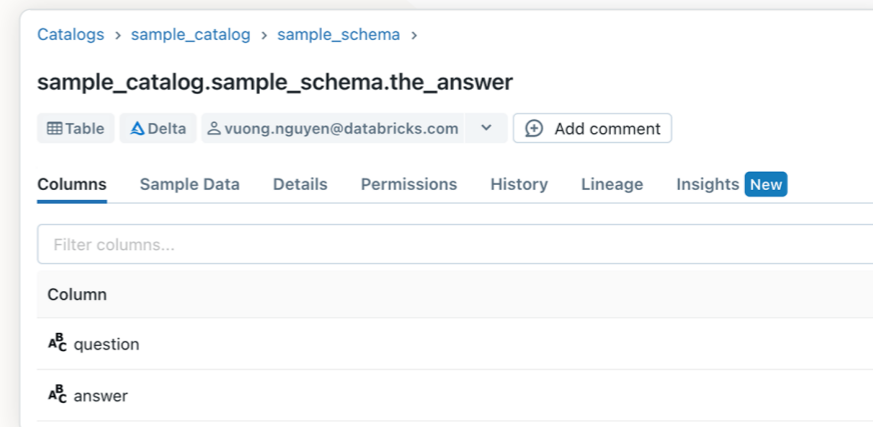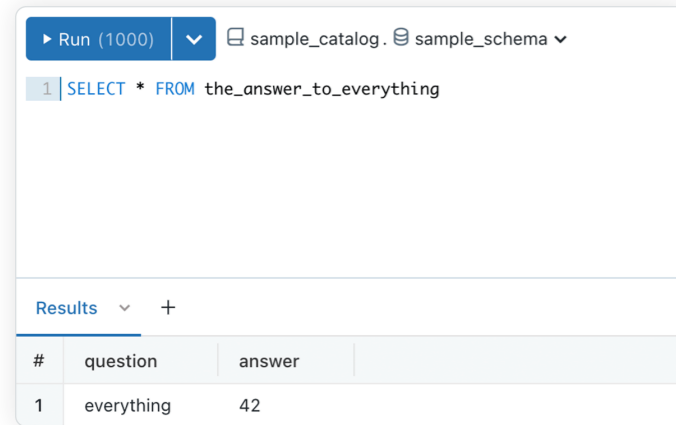
Ownership of the table
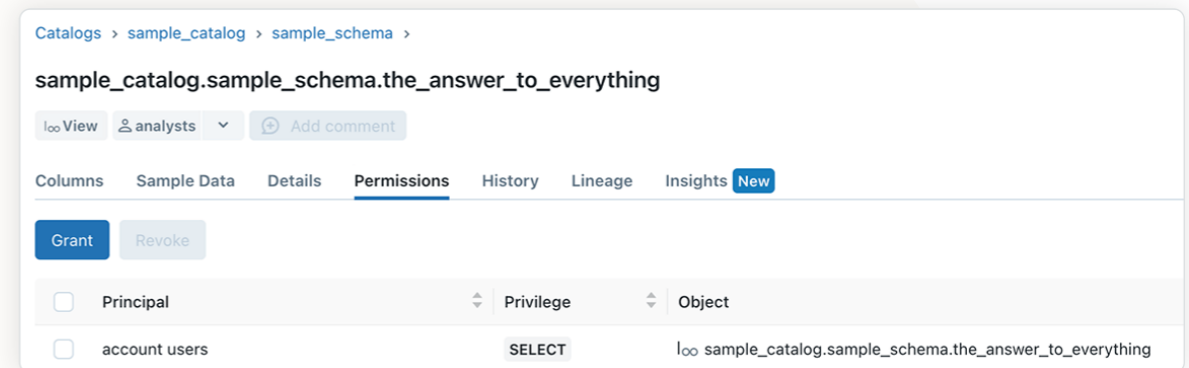
Permission page showing a view owned by analysts group, and account users group having SELECT permission
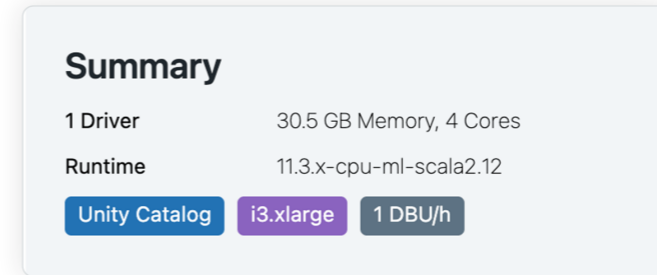
Query page showing the result of the answer to everything

Cluster summary indicating Unity Catalog support

Example 4 – No more data leakage

Thanks to axiom 6, data owners can be certain that there will be no unauthorized access to their data due to cluster misconfiguration. Any cluster that is not configured with the correct access mode will not be able to access data in Unity Catalog.

Users can check that their clusters can access Unity Catalog data thanks to this handy tooltip on the Create Clusters page

Now that data owners can understand the data privilege model and access control, they can leverage Unity Catalog to simplify access policy management at scale.

There are upcoming features that will further empower data administrators |and owners to author even more complex access policy:

- **Row filtering and column masking:** Use standard SQL functions to define row filters and column masks, allowing fine-grained access controls on rows and columns.

- **Attribute Based Access Controls:** Define access policies based on tags (attributes) of your data assets.

databricks

SECTION 4.7

Onboarding Your New AI/BI Genie

by Chao Cai and Richard Tomlinson

At Databricks, we want to make data and AI accessible to everyone on the planet. This is why we're building solutions like AI/BI that make it possible for business users, even those who don't speak SQL or write code, to get trusted insights quickly and accurately.

At this point, dashboards are part of most people's daily lives. On the other hand, AI/BI Genie is a newer experience, one that many people might not be familiar with yet. With Genie, users can self-serve and obtain answers to questions not addressed in their dashboards without having to learn how to use BI tools or rely on expert practitioners to create insights for them. With this backdrop, we figured it might be worthwhile to take a look at a day in the life of a Genie so that you can get to know how to use it and what to expect.

New hire orientation

Some folks have made the analogy that the process of creating a Genie is akin to hiring a new analyst for your data team. This new hire is smart and hungry to learn but is a tad robotic and has zero experience with any of your internal terminology, metrics, definitions and assumptions. Your new Genie is well-read in public knowledge but very ignorant of company-internal and team-internal knowledge.

The first step is for someone to give your new hire some much-needed orientation. This task likely falls to you, the experienced analyst who has been fielding all of the team's questions, perhaps drowning in them and looking for some help.

The Genie setup page is your guide to what this new hire needs to know:

New
Create a new space by giving it a name and define which tables make sense for the use case.

Title

Description
Describe what data is available in this space and what type of questions users can ask.

Default warehouse

Tables
Choose tables to use for answering questions in the space. It is best to keep the scope for each space as small as possible. Data access is governed with the viewers Unity Catalog permissions.

Catalog	Schema	Table

Sample questions
Sample questions will be presented in new chat windows for users to ask the Space.

E.g. What is our annual revenue?

Run As Setting
When sharing the space with other users, select whether to embed the publishing user's credentials, or to use the viewer's credentials

⦿ Viewer
○ Embedded Credentials

Cancel   Save

databricks

For this screen, the goal is to provide the basic information needed to get Genie going:

- **Title and description:** What is the scope and domain of questions you expect Genie to know how to answer? This is visible to anyone browsing and searching across all of the Genie spaces they can access, so it's important to make this clear and succinct.

- **Default warehouse:** What compute will this Genie use to run SQL queries for you?

- **Tables:** What tables are in scope for this Genie to use to answer questions? Genie will read up on all of the comments in each table and in each of its columns and use these to inform its answers.

- **Sample questions:** Optionally, you can specify some example questions you want to present to people who are talking to this Genie if they're not sure what to ask.

Once you've chosen sensible values for these fields, click Save and your new Genie is ready for action.

Well, not quite. If you were training a new hire, you'd probably want to check their knowledge first before letting them go off to answer your team's hardest questions.

To ensure your new hire has the best possible start in their new career you'll want to bootstrap their knowledge with some basic information. If you've invested in some basic data governance best practices you'll ensure they get a flying start. This is where Unity Catalog can help. Make sure you've created some useful comments in your tables and columns, and take the time to create primary/foreign key relationships between the tables you want to give to Genie. This basic information will ensure Genie has a lot of basic knowledge on its very first day.

The next step is to ask Genie a few things that you'd expect your team to ask in daily practice.

In this first example, Genie easily gets the question correct.

In Catalog Explorer (above), if we look at the table metadata, we see the column comments provide enough information for Genie to get the answer right. Yay!

In this second example, Genie guessed wrong because the most obvious definition it infers from the column name for average price turns out to be wrong.

When we view the underlying SQL we can see that Genie picked the wrong column for price. Genie should've picked the ListPrice column, not UnitPrice.

We can fix this by giving the desired column a useful clarifying comment in Catalog Explorer:

With this update, Genie will now produce the right answer.

The user asks Genie the exact same question as before.

This time Genie uses the correct column, ListPrice, in its query.

In the next example, Genie won't even try to guess because it knows that it lacks an important context: how a particular team-specific metric is defined. Without this context, Genie has no way of deducing the right answer.

So let's teach Genie this metric. We can provide the metric directly to Genie and see what happens.

We simply tell Genie the definition of customer churn directly in the UI and run the prompt.

We can see that Genie has understood the prompt and is able to query the data correctly.

Now that we've confirmed that Genie learned how to calculate this new metric correctly, we can tell it to remember this going forward by clicking "Add as instruction."
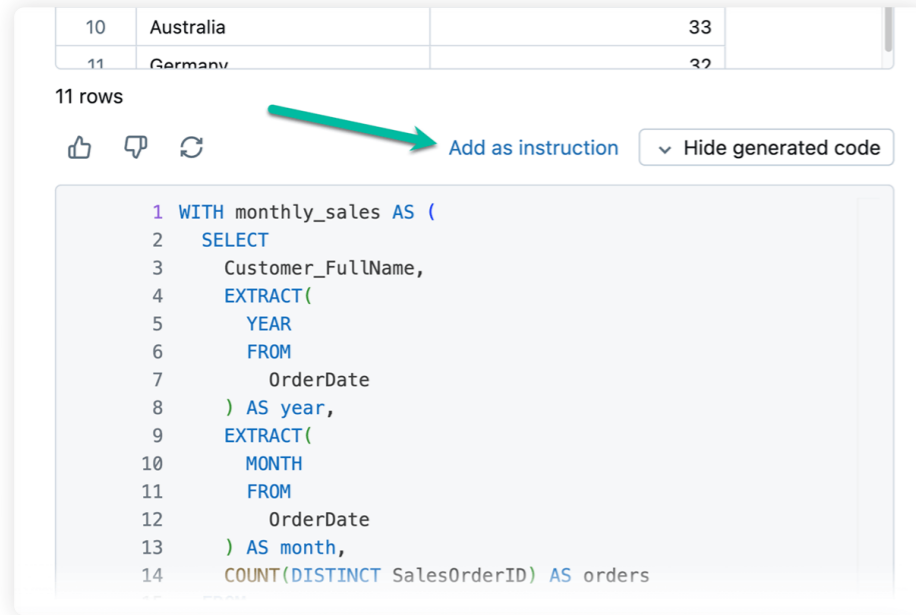
This action saves the instruction into the Instructions section, which contains all the things you want Genie to consider as it does its job. This instruction can be found in Example SQL Queries, but you can also manually update any of the instructions.
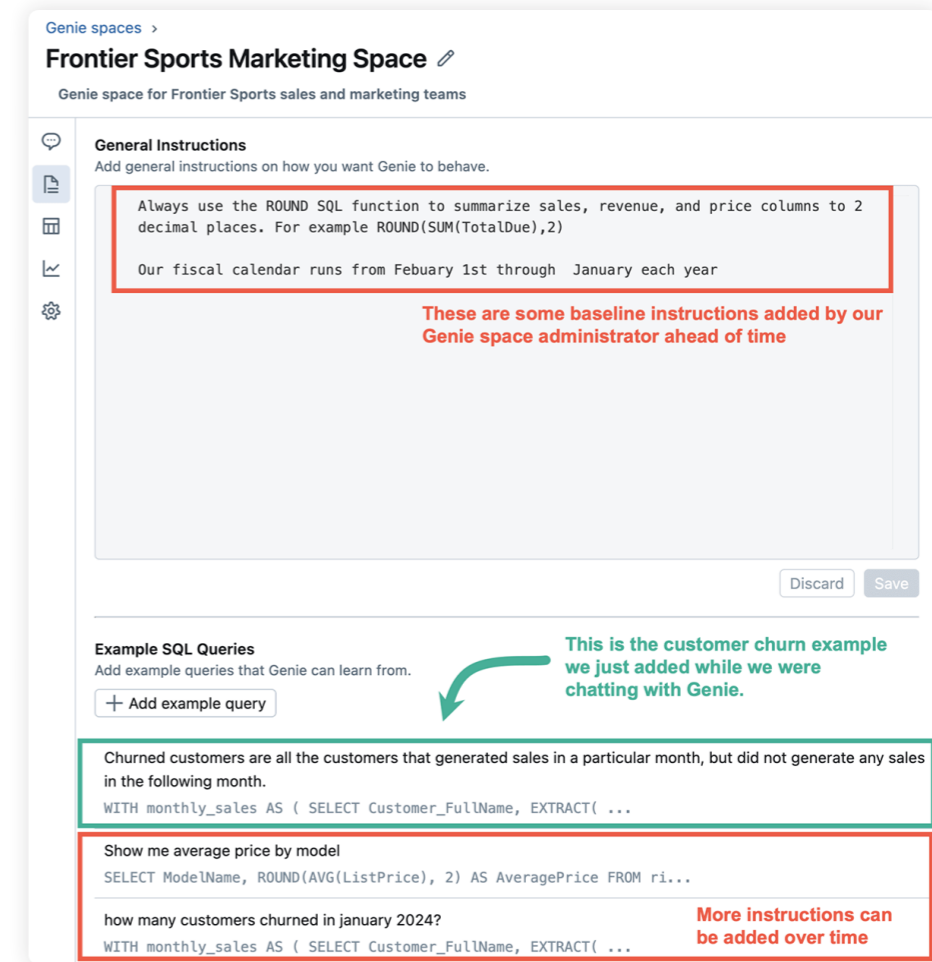
Before releasing your Genie space into the wild, we recommend you add as many of these types of instructions as possible for the known/anticipated questions and business semantics your users will need. You can do this by saving instructions as you chat or by adding them directly to the Instructions section in your space. The more you train your Genie, the smarter it gets.

databricks

Once you've tested Genie enough to be confident that it's providing useful and correct answers to your team's anticipated questions, it's time to let your team talk to the new hire.

Click Share to tell Genie who can talk to it. Depending on who this Genie is supposed to answer questions for, you can choose individual people, a group representing your whole team or the entire company.

You should also tell your team about what to expect from Genie, since they've probably never worked with such a robotic teammate before. Remember that they probably can't speak SQL, so they might not be able to tell whether Genie is actually giving them the right answer or just making a guess that's actually wrong. You can tell your team about what questions you expect Genie to answer correctly. Be sure to let them know that if they have a mission-critical question but aren't sure whether Genie is answering correctly, they should check with you before using it.

Getting to work

So now that you've unleashed your new Genie teammate, let's see how a member of your team interacts with it.

Similar to when you tested Genie, anyone on the team can start asking questions and getting answers back.

Your users can give feedback by clicking the thumbs-up/thumbs-down buttons after seeing Genie's response. These votes show up in the Monitoring section, where you can see the questions your users are asking and identify new things you need to teach Genie to improve it.

Learn more about Genie

As you can see, a main goal of Genie is to make it possible for anyone on any team to get the insights they're looking for without overtaxing or waiting for the more technical members of their team to provide answers. We hope you've found these examples helpful and hope that you'll give Genie a try. For a deeper look at Genie, check out the Databricks documentation around Genie, especially the best practices we've summarized. Please let us know if you have feedback. We look forward to learning about your experiences with Genie.

Here you can view all of the questions that have been asked and any thumbs-up/thumbs-down feedback from your team.

Look for questions that you hadn't anticipated earlier and aren't sure whether Genie might answer correctly. For each question, you can look at how Genie answered to check whether it's correct. If it isn't correct, this is an opportunity to hop into a chat session of your own to teach Genie the right answer.

You can also filter for "thumbs down" interactions to check for two different scenarios. First, Genie might've answered incorrectly, so you'll need to teach Genie something it doesn't know. Second, you might think that Genie answered correctly, but your teammate talking to Genie might have different assumptions than you do, leading them to conclude that Genie isn't answering correctly or usefully. It'll be helpful to clear up those differences in assumptions.

SECTION 4.8

How to Embed AI/BI Dashboards Into Your Websites and Applications

by Eason Gao

We're thrilled to announce that embedding for AI/BI Dashboards is now available. Embedding enables you to seamlessly integrate Databricks AI/BI Dashboards into third-party tools such as your internal websites and applications. This integration simplifies access to data insights for your team, allowing them to interact with the information without disrupting their workflow.
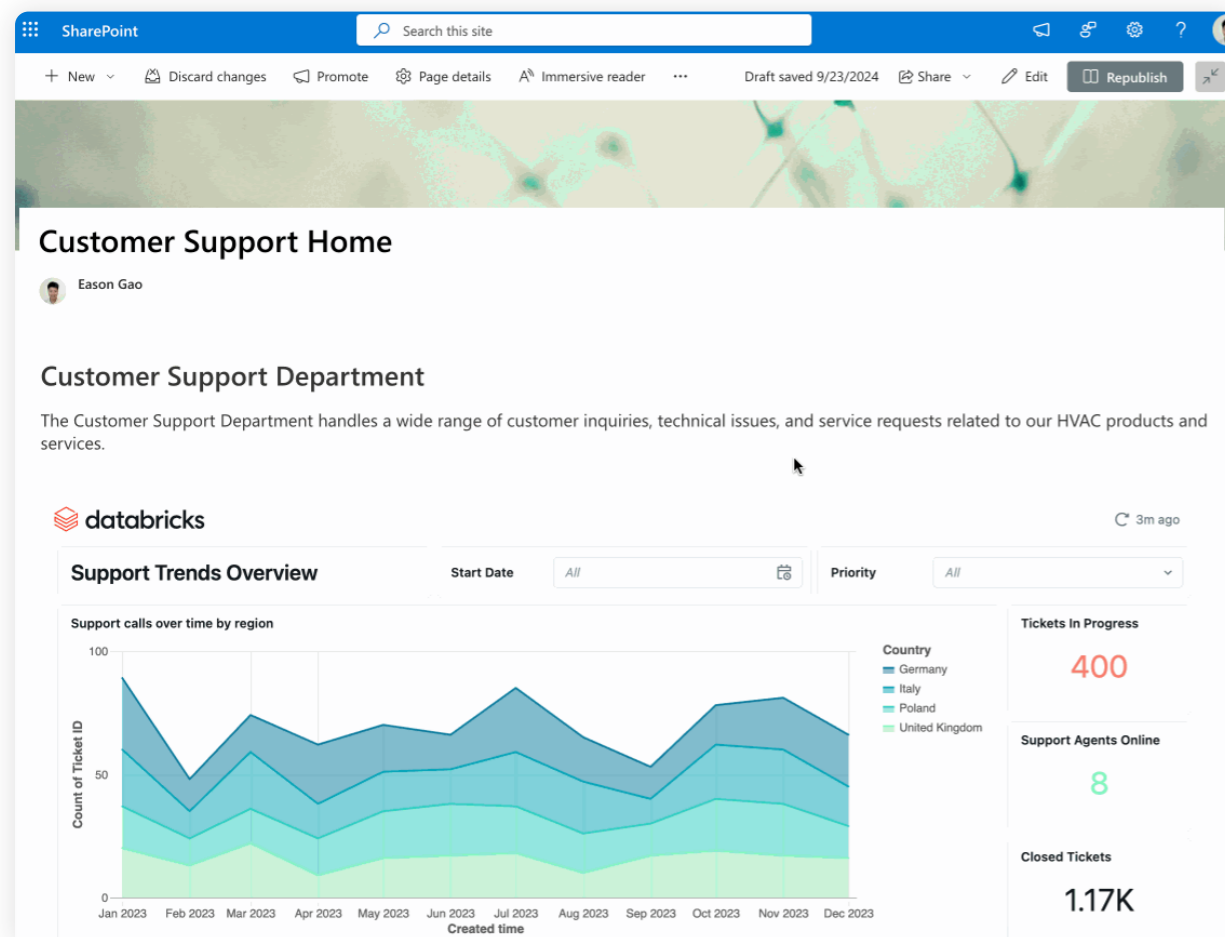
What is dashboard embedding?

Dashboard embedding enables Databricks AI/BI Dashboards to be displayed within other platforms using iframes. This means that users can view dashboards without having to leave the applications they're already using, streamlining workflows and enhancing productivity.

Key features

- **Integration Flexibility:** You can embed dashboards anywhere iframes are welcome. This list encompasses common productivity tools like Confluence, Salesforce and SharePoint. This flexibility ensures that data insights are accessible in the tools your teams use regularly.

- **Secure Access:** To maintain security, only authorized users can access the dashboard in the embedded surface. To authorize a user, make sure they're part of your Databricks account and have been shared the dashboard.

- **Optimized Performance:** Embedded dashboards benefit from the same client-side filtering and caching techniques as in-product dashboards, ensuring snappy performance

How to use dashboard embedding

To start embedding dashboards, follow these steps:

1. **User Provisioning:** Ensure that all users who need to view embedded dashboards are first registered with Databricks. Do this by adding users to the Databricks account. They don't need to be granted access to any workspaces. Users without workspace access can view shared AI/BI Dashboards but they can't edit them or access other workspace content like Workflows, ML experiments, etc.

   Account-level SCIM provisioning can simplify registering authorized dashboard viewers with Databricks. See this blog post to learn more about sharing AI/BI Dashboards with everyone in your organization.
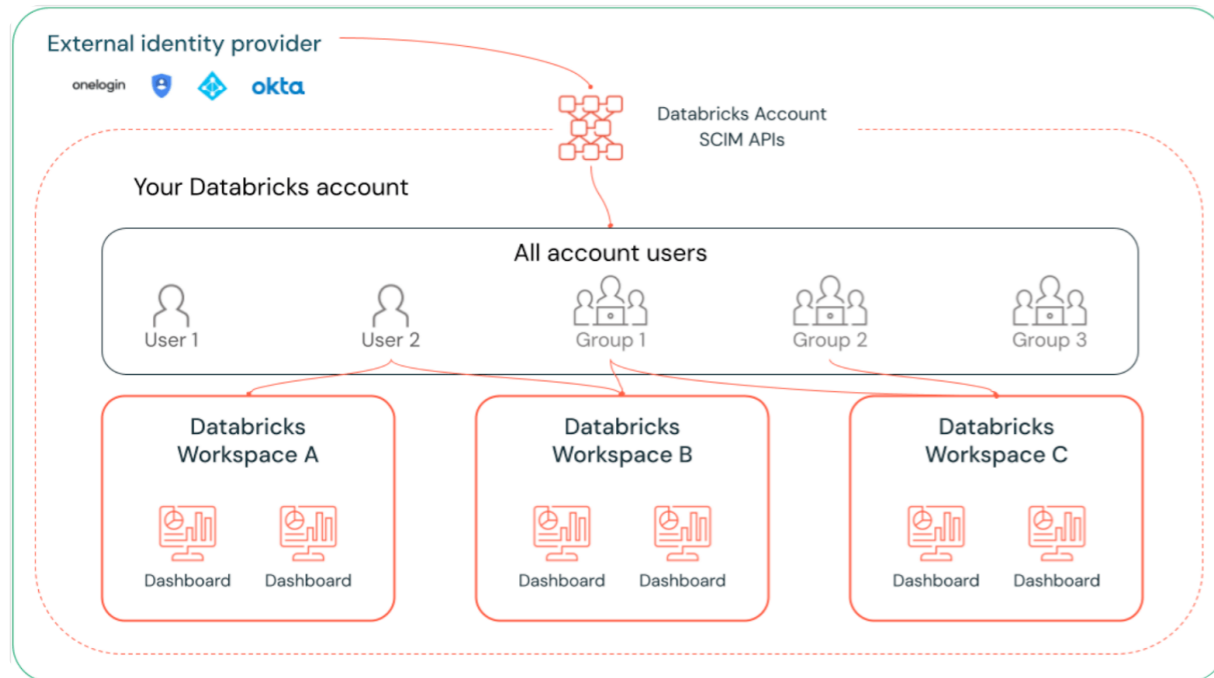
2. **Define Allowed Domains:** In workspace settings, ensure that a list of allowed domains where dashboards may be embedded is specified. This allow list is empty by default so this feature is effectively disabled until a workspace admin enables it by specifying allowed domains.
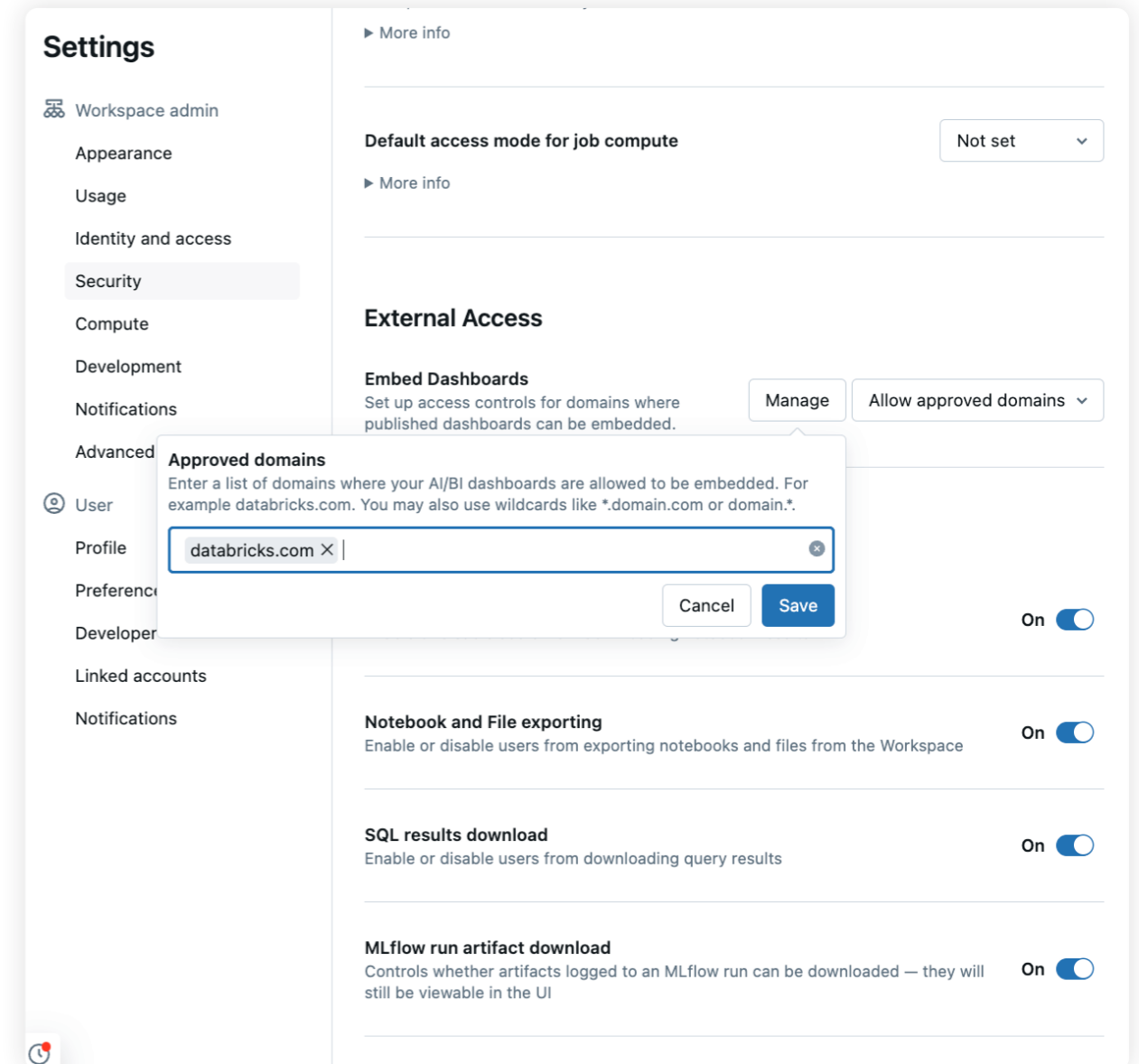
3. **Dashboard Creation:** Design and publish your dashboards within Databricks. Share your dashboard with all users and/or groups who should have access. If you're sharing to account-level users, publish with embedded credentials to ensure that they can access the dashboard.

4. **Embedding:** Copy the iframe embed code from the dashboard share dialog and insert it into your desired platform.

Publish Domestic US Flight Delays

General settings

A published dashboard accesses live data, running queries with your embedded or viewing user's credentials. If you are sharing with users in your organization without Databricks workspace access, select embed credentials so that queries can run and charts can render.

Embed credentials (default)

✓ ⓘ Embed credentials (default)

ⓘ **Don't** embed credentials

of the published dashboard will use your data and warehouse permissions to run queries.

People with access

Users with at least "Can View" permissions can see the published dashboard. Click here to change user permissions.

eason.gao@databricks.com, Admins, account users

🔗 Copy link   Unpublish   Publish

Example for embedding in SharePoint

Example for embedding in Confluence

Conclusion

Embedding provides an opportunity for you to integrate data–driven insights seamlessly into your organization's existing tools. We're excited to see how you and your teams will leverage this alongside AI/BI's many other capabilities to transform your business processes. We encourage you to dive in and start creating your own dashboards today. To get started, explore our AI/BI Dashboards documentation and follow our tutorial to build your first dashboard. To see AI/BI Dashboards and Genie in action, check out our demo and take the product tour.

Your feedback is invaluable as we continue to refine and expand our AI/BI Dashboards. We look forward to hearing your thoughts and suggestions. Happy dashboarding!

databricks

Analytics Use Cases on Databricks

databricks

How to Build a Marketing Analytics Solution Using Fivetran and dbt on Databricks

by Tahir Fayyaz, Bilal Aslam and Robert Saxby

Marketing teams use many different platforms to drive marketing and sales campaigns, which can generate a significant volume of valuable but disconnected data. Bringing all of this data together can help to drive a large return on investment, as shown by Publicis Groupe who were able to increase campaign revenue by as much as 50%.

Databricks, which unifies data warehousing and AI use cases on a single platform, is the ideal place to build a marketing analytics solution: we maintain a single source of truth, and unlock AI/ML use cases. We also leverage two Databricks partner solutions, Fivetran and dbt, to unlock a wide range of marketing analytics use cases including churn and lifetime value analysis, customer segmentation, and ad effectiveness.

Fivetran allows you to easily ingest data from 50+ marketing platforms into Delta Lake without the need for building and maintaining complex pipelines. If any of the marketing platforms' APIs change or break, Fivetran will take care of updating and fixing the integrations so your marketing data keeps flowing in.

dbt is a popular open source framework that lets lakehouse users build data pipelines using simple SQL. Everything is organized within directories, as plain text, making version control, deployment, and testability simple. Once the data is ingested into Delta Lake, we use dbt to transform, test and document the data. The transformed marketing analytics data mart built on top of the ingested data is then ready to be used to help drive new marketing campaigns and initiatives.
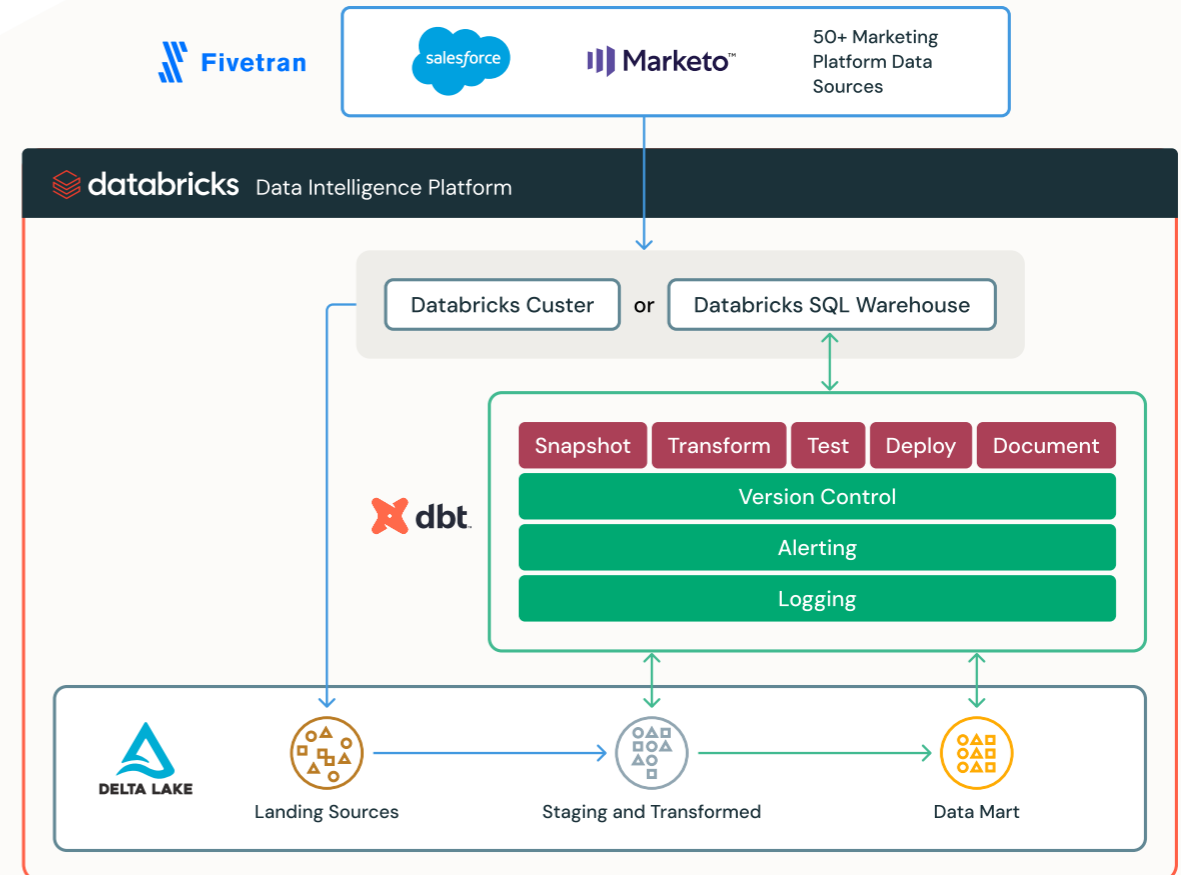

Fivetran and dbt can read and write to Delta Lake using a Databricks cluster or Databricks SQL warehouse

Both Fivetran and dbt are a part of Databricks Partner Connect, a one-stop portal to discover and securely connect data, analytics and AI tools directly within the Databricks Platform. In just a few clicks you can configure and connect these tools (and many more) directly from within your Databricks workspace.

How to build a marketing analytics solution

In this hands-on demo, we will show how to ingest Marketo and Salesforce data into Databricks using Fivetran and then use dbt to transform, test, and document your marketing analytics data model.

All the code for the demo is available on Github in the workflows-examples repository.
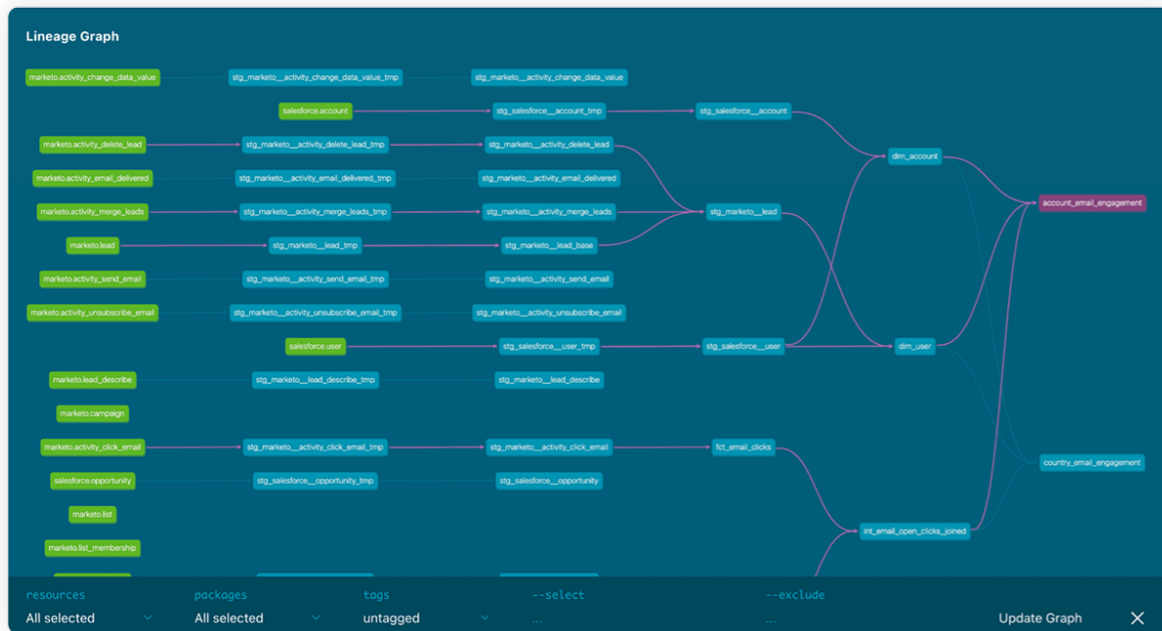
dbt lineage graph showing data sources and models

The final dbt model lineage graph will look like this. The Fivetran source tables are in green on the left and the final marketing analytics models are on the right. By selecting a model, you can see the corresponding dependencies with the different models highlighted in purple.
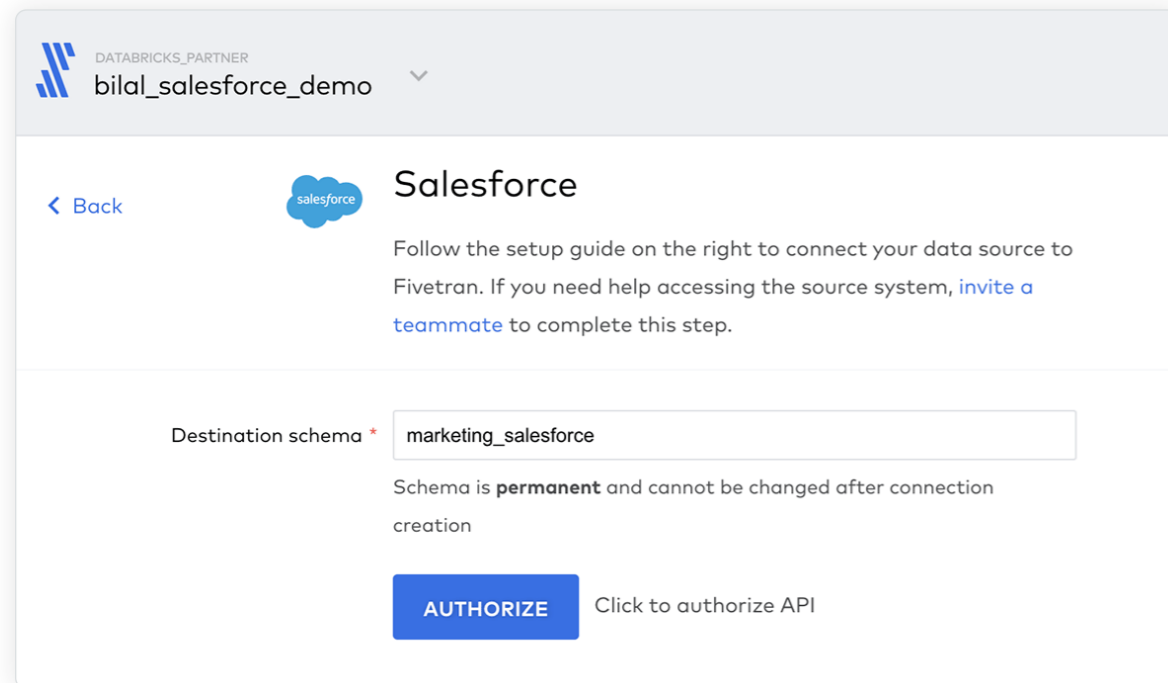
Data ingestion using Fivetran
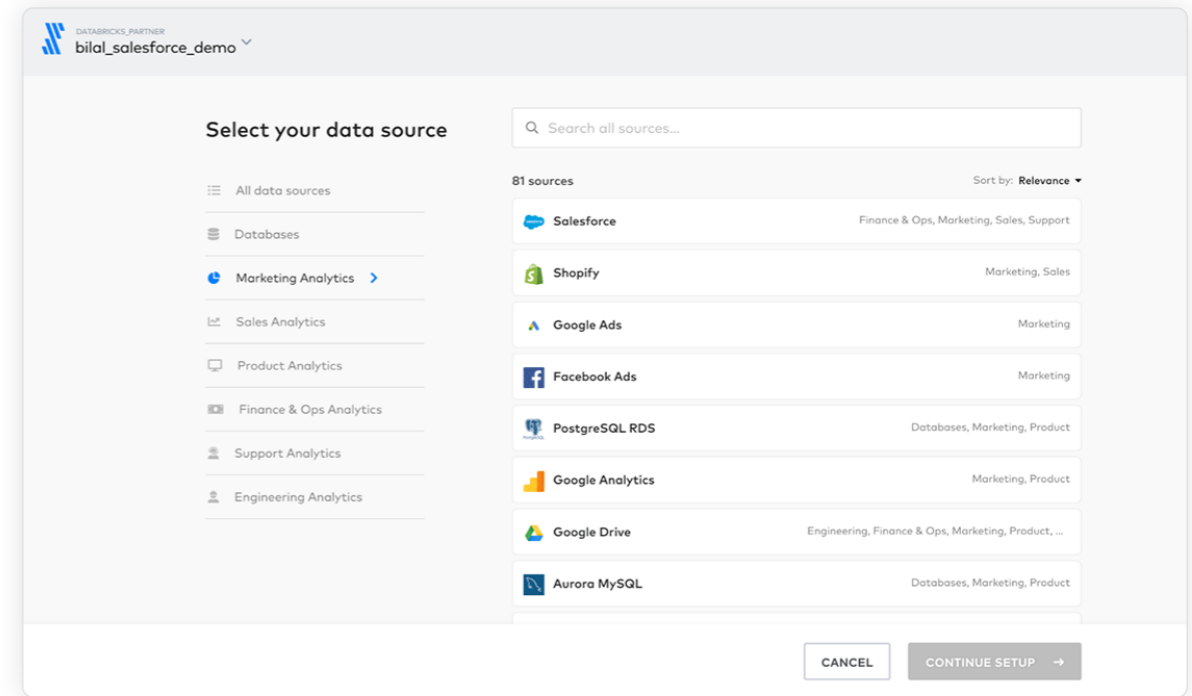
Fivetran has many marketing analytics data source connectors
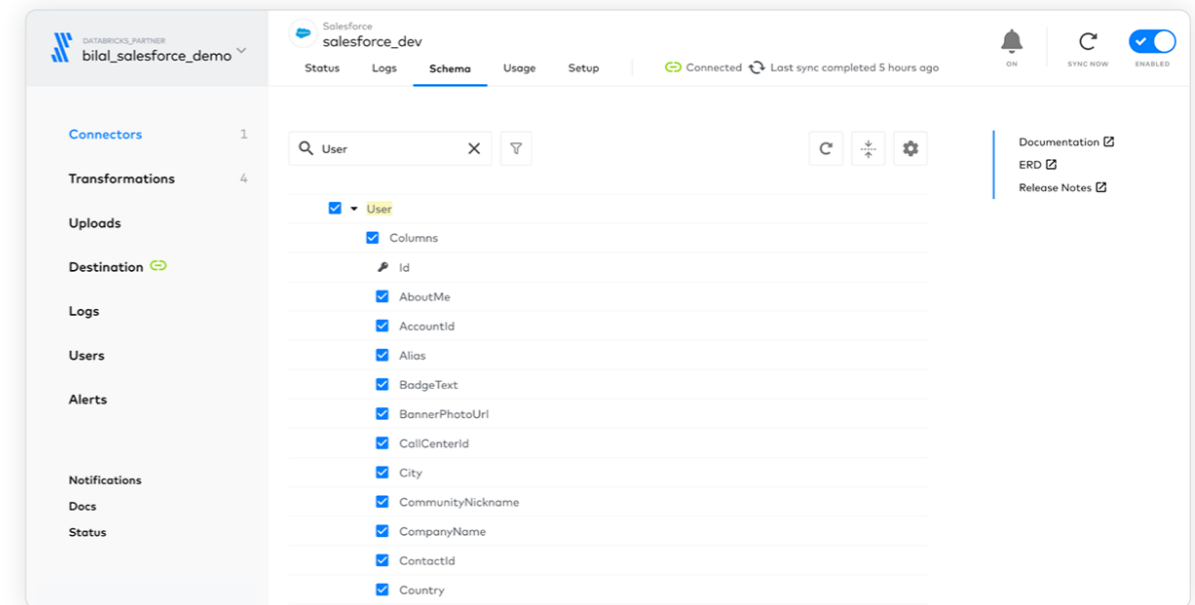
databricks

Create new Salesforce and Marketo connections in Fivetran to start ingesting the marketing data into Delta Lake. When creating the connections Fivetran will also automatically create and manage a schema for each data source in Delta Lake. We will later use dbt to transform, clean and aggregate this data.

For the demo name the schemas that will be created in Delta Lake marketing_salesforce and marketing_marketo. If the schemas do not exist Fivetran will create them as part of the initial ingestion load.

Define a destination schema in Delta Lake for the Salesforce data source

Select which data source objects to synchronize as Delta Lake tables

You can then choose which objects to sync to Delta Lake, where each object will be saved as individual tables. Fivetran also makes it simple to manage and view what columns are being synchronized for each table:
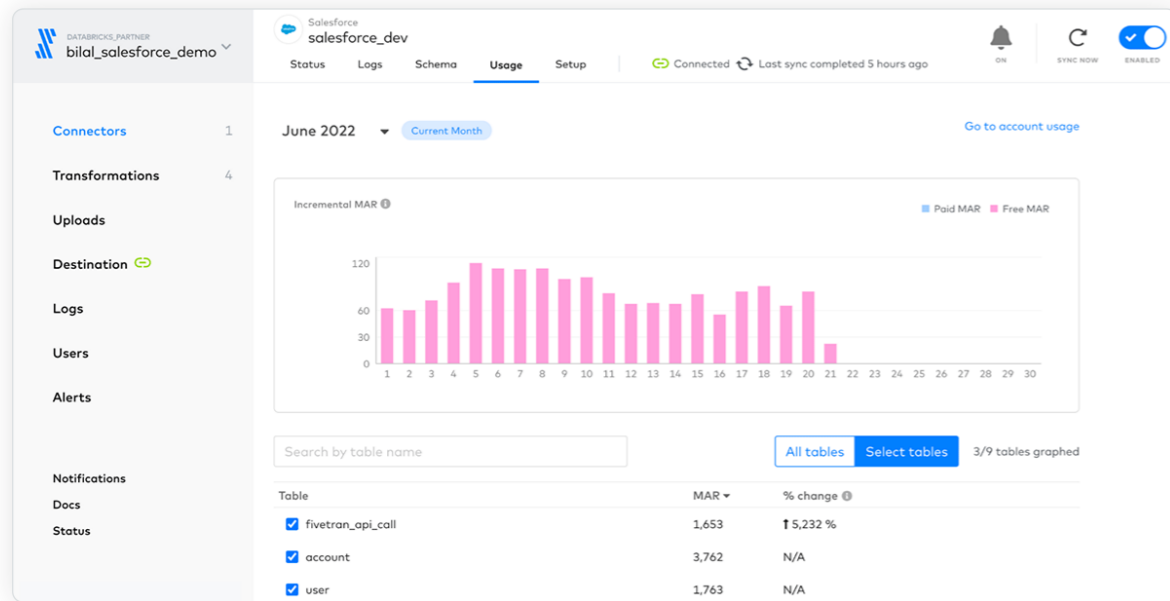
Fivetran monitoring dashboard to monitor monthly active rows synchronized

Additionally, Fivetran provides a monitoring dashboard to analyze how many monthly active rows of data are synchronized daily and monthly for each table, among other useful statistics and logs.

Data modeling using dbt

Now that all the marketing data is in Delta Lake, you can use dbt to create your data model by following these steps

Setup dbt project locally and connect to Databricks SQL

Set up your local dbt development environment in your chosen IDE by following the set-up instructions for dbt Core and dbt-databricks.

Scaffold a new dbt project and connect to a Databricks SQL Warehouse using dbt init, which will ask for following information.

```
$ dbt init
Enter a name for your project (letters, digits, underscore):
Which database would you like to use?
[1] databricks
[2] spark

Enter a number: 1
host (yourorg.databricks.com):
http_path (HTTP Path):
token (dapiXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX):
schema (default schema that dbt will build objects in):
threads (1 or more) [1]:
```

Once you have configured the profile you can test the connection using:

```
$ dbt debug
```

Install Fivetran dbt model packages for staging

The first step in using the Marketo and Salesforce data is to create the tables as sources for our model. Luckily, Fivetran has made this easy to get up and running with their pre-built Fivetran dbt model packages. For this demo, let's make use of the `marketo_source` and `salesforce_source` packages.

To install the packages just add a `packages.yml` file to the root of your dbt project and add the `marketo-source`, `salesforce-source` and the `fivetran-utils` packages:

```
1   packages:
2     - package: dbt-labs/spark_utils
3       version: 0.3.0
4     - package: fivetran/marketo_source
5       version: [">=0.7.0", "=0.4.0", "
6
7   To download and use the packages run
```

```
1   $ dbt deps
```

You should now see the Fivetran packages installed in the packages folder.

Update dbt_project.yml for Fivetran dbt models

There are a few configs in the `dbt_project.yml` file that you need to modify to make sure the Fivetran packages work correctly for Databricks.

The `dbt_project.yml` file can be found in the root folder of your dbt project.

spark_utils overriding dbt_utils macros

The Fivetran dbt models make use of macros in the `dbt_utils` package but some of these macros need to be modified to work with Databricks which is easily done using the `spark_utils` package.

It works by providing shims for certain `dbt_utils` macros which you can set using the dispatch config in the `dbt_project.yml` file and with this dbt will first search for macros in the `spark_utils` package when resolving macros from the `dbt_utils` namespace.

```
1   dispatch:
2     - macro_namespace: dbt_utils
3       search_order: ['spark_utils', 'dbt_utils']
```

Variables for the marketo_source and salesforce_source schemas

The Fivetran packages require you to define the catalog (referred to as database in dbt) and schema of where the data lands when being ingested by Fivetran.

Add these variables to the `dbt_project.yml` file with the correct catalog and schema names. The default catalog is `hive_metastore` which will be used if `_database` is left blank. The schema names will be what you defined when creating the connections in Fivetran.

```
1   vars:
2     marketo_source:
3       marketo_database: # leave blank to use the default hive_metastore catalog
4       marketo_schema: marketing_marketo
5     salesforce_source:
6       salesforce_database: # leave blank to use the default hive_metastore catalog
7       salesforce_schema: marketing_salesforce
```

databricks

Target schema for Fivetran staging models

To avoid all the staging tables that are created by the Fivetran source models being created in the default target schema it can be useful to define a separate staging schema.

In the `dbt_project.yml` file add the staging schema name and this will then be suffixed to the default schema name.

```
1  models:
2    marketo_source:
3      +schema: your_staging_name # leave blank to use the default target_schema
4    salesforce_source:
5      +schema: your_staging_name # leave blank to use the default target_schema
```

Based on the above, if your target schema defined in `profiles.yml` is `mkt_analytics`, the schema used for `marketo_source` and `salesforce_source` tables will be `mkt_analytics_your_staging_name`.

Disable missing tables

At this stage you can run the Fivetran model packages to test that they work correctly.

```
1  dbt run -select marketo_source
```

```
1  dbt run -select marketo_source
```

If any of the models fail due to missing tables, because you chose not to sync those tables in Fivetran, then in your source schema you can disable those models by updating the `dbt_project.yml` file.

For example if the email bounced and email template tables are missing from the Marketo source schema you can disable the models for those tables by adding the following under the models config:

```
1   models:
2     marketo_source:
3       +schema: your_staging_name
4       tmp:
5         stg_marketo__activity_email_bounced_tmp:
6           +enabled: false
7         stg_marketo__email_template_history_tmp:
8           +enabled: false
9       stg_marketo__activity_email_bounced:
10        +enabled: false
12      stg_marketo__email_template_history:
13        +enabled: false
```

databricks

Developing the marketing analytics models
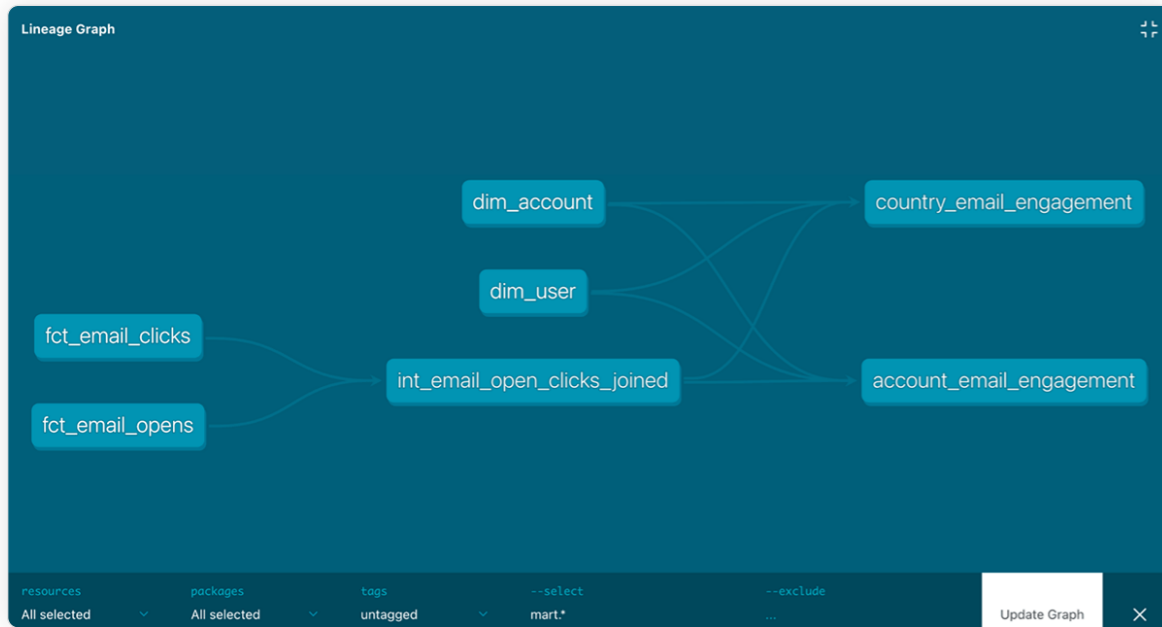
dbt lineage graph showing the star schema and aggregate tables data model

Now that the Fivetran packages have taken care of creating and testing the staging models you can begin to develop the data models for your marketing analytics use cases which will be a star schema data model along with materialized aggregate tables.

For example, for the first marketing analytics dashboard, you may want to see how engaged certain companies and sales regions are by the number of email campaigns they have opened and clicked.

To do so, you can join Salesforce and Marketo tables using the Salesforce user email, Salesforce `account_id` and Marketo `lead_id`.

The models will be structured under the mart folder in the following way.

```
1    marketing_analytics_demo
2    |-- dbt_project.yml
3    |-- packages.yml
4    |-- models
5        |-- mart
6            |-- core
7            |-- intermediate
8            |-- marketing_analytics
```

You can view the code for all the models on Github in the /models/mart directory and below describes what is in each folder along with an example.

Core models

The core models are the facts and dimensions tables that will be used by all downstream models to build upon.

The dbt SQL code for the `dim_user` model:

```
1    with salesforce_users as (
2        select
3            account_id,
4            email
5        from {{ ref('stg_salesforce__user') }}
6        where email is not null and account_id is not null
7    ),
8    marketo_users as (
9        select
10           lead_id,
12           email
13       from {{ ref('stg_marketo__lead') }}
14   ),
15   joined as (
16       select
17           lead_id,
18           account_id
19       from salesforce_users
20           left join marketo_users
21           on salesforce_users.email = marketo_users.email
22   )
23
24   select * from joined
```

databricks

You can also add documentation and tests for the models using a yaml file in the folder.

There are 2 simple tests in the `core.yml` file that have been added

```
1    version: 2
2
3    models:
4     - name: dim_account
5       description: "The Account Dimension Table"
6       columns:
7         - name: account_id
8           description: "Primary key"
9           tests:
10            - not_null
12    - name: dim_user
13      description: "The User Dimension Table"
14      columns:
15        - name: lead_id
16          description: "Primary key"
17          tests:
18            - not_null
```

Intermediate models

Some of the final downstream models may rely on the same calculated metrics and so to avoid repeating SQL you can create intermediate models that can be reused.

The dbt SQL code for `int_email_open_clicks_joined` model:

```
1    with opens as (
2          select *
3          from {{ ref('fct_email_opens') }}
4    ), clicks as (
5          select *
6          from {{ ref('fct_email_clicks') }}
7    ), opens_clicks_joined as (
8
9      select
10       o.lead_id as lead_id,
12       o.campaign_id as campaign_id,
13       o.email_send_id as email_send_id,
14       o.activity_timestamp as open_ts,
15       c.activity_timestamp as click_ts
16     from opens as o
17       left join clicks as c
18       on o.email_send_id = c.email_send_id
19       and o.lead_id = c.lead_id
20
21    )
22
23    select * from opens_clicks_joined
```

Marketing Analytics models

These are the final marketing analytics models that will be used to power the dashboards and reports used by marketing and sales teams.

The dbt SQL code for `country_email_engagement` model:

```sql
with accounts as (
        select
            account_id,
            billing_country
            from {{ ref('dim_account') }}
), users as (
        select
            lead_id,
            account_id
            from {{ ref('dim_user') }}
), opens_clicks_joined as (

    select * from {{ ref('int_email_open_clicks_joined') }}

), joined as (

        select *
        from users as u
        left join accounts as a
        on u.account_id = a.account_id
        left join opens_clicks_joined as oc
        on u.lead_id = oc.lead_id

)

select
        billing_country as country,
        count(open_ts) as opens,
        count(click_ts) as clicks,
        count(click_ts) / count(open_ts) as click_ratio
from joined
group by country
```

Run and test dbt models

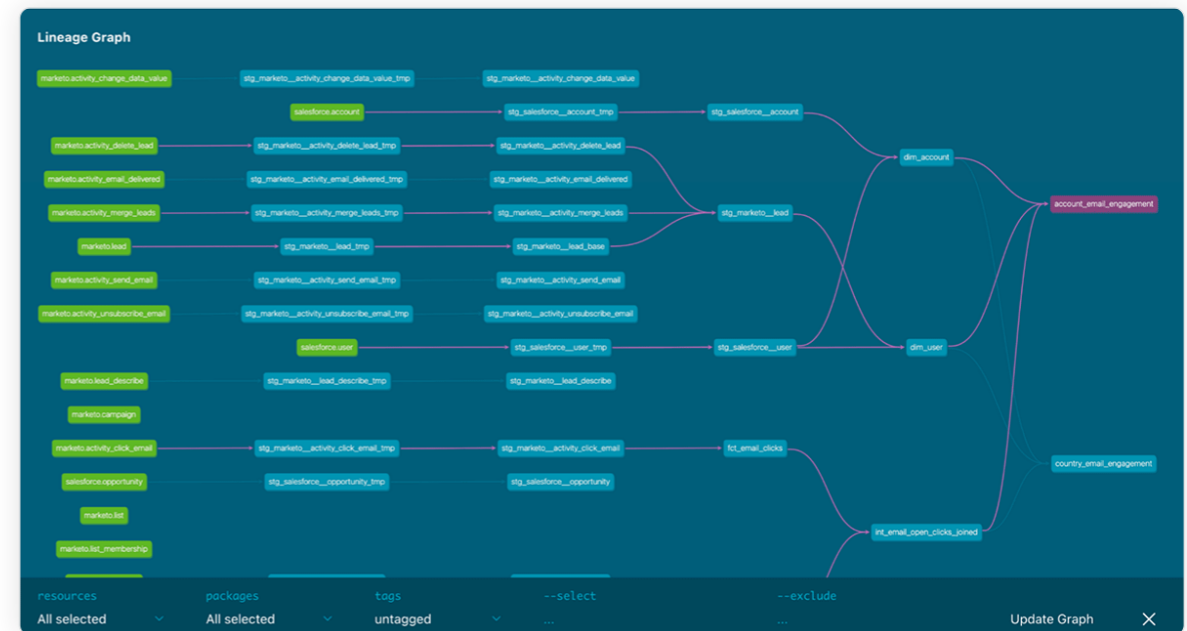Now that your model is ready you can run all the models using

```
dbt run
```

And then run the tests using

```
dbt test
```

View the dbt docs and lineage graph

dbt lineage graph for the marketing analytics model

Once your models have run successfully you can generate the docs and lineage graph using

```
$ dbt docs generate
```

To then view them locally run

```
$ dbt docs serve
```

Deploying dbt models to production

Once you have developed and tested your dbt model locally you have multiple options for deploying into production one of which is the new dbt task type in Databricks Workflows (private preview).

Your dbt project should be managed and version controlled in a Git repository. You can create a dbt task in your Databricks Workflows job pointing to the Git repository.
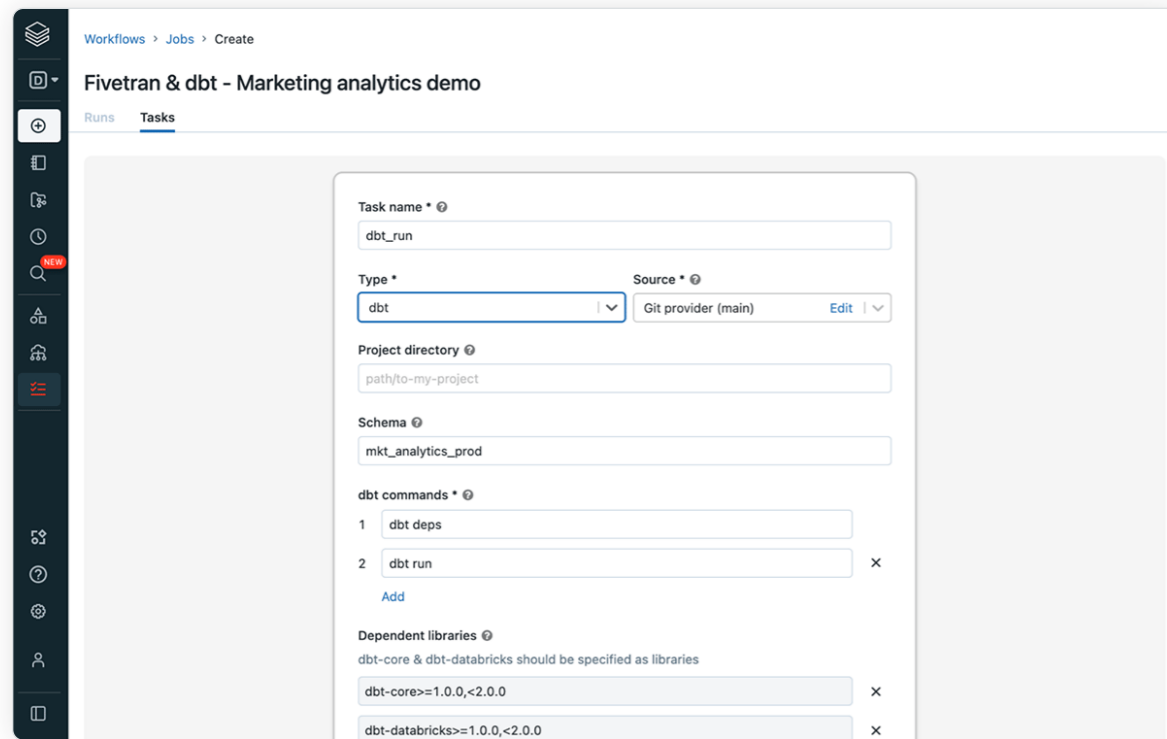
Using a dbt task type in Databricks Workflows to orchestrate dbt

As you are using packages in your dbt project the first command should be dbt deps followed by dbt run for the first task and then dbt test for the next task.
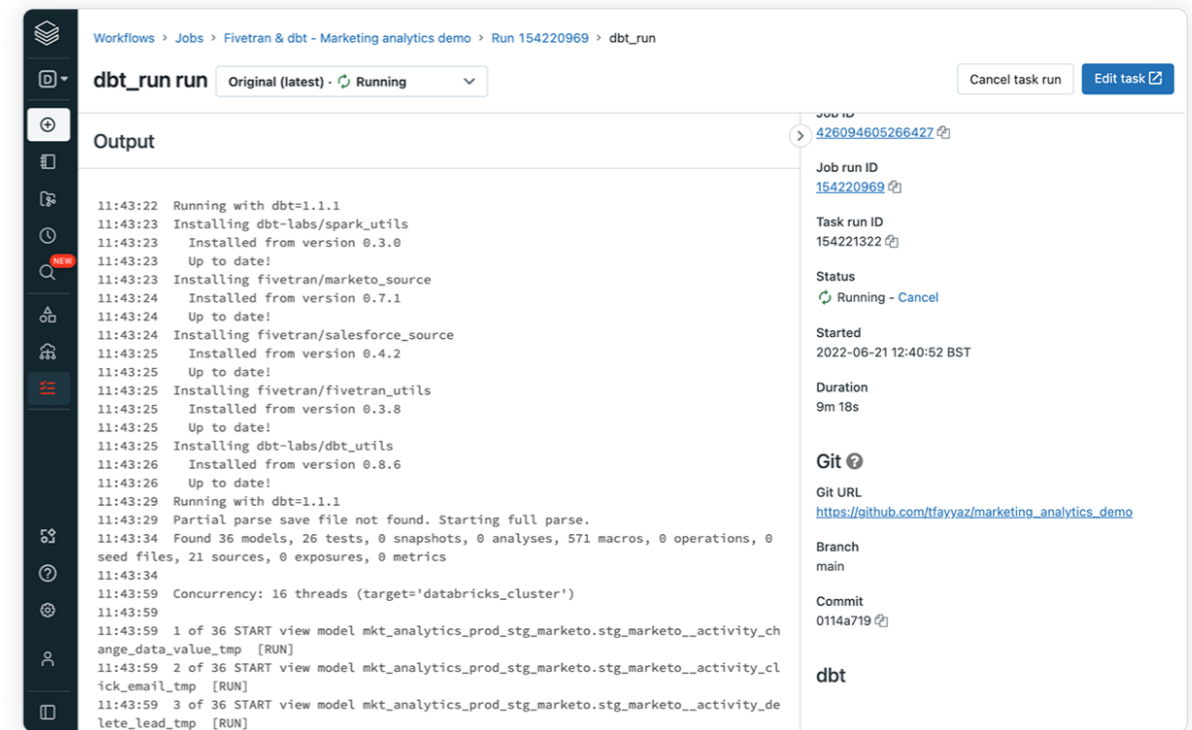
Viewing the dbt logs for each dbt run

For each run you can see the logs for each dbt command helping you debug and fix any issues.

databricks

Powering your marketing analytics with Fivetran and dbt

As shown here, using Fivetran and dbt alongside the Databricks Data Intelligence Platform allows you to easily build a powerful marketing analytics solution that is easy to set up, manage and flexible enough to suit any of your data modeling requirements.

To get started with building your own solution, visit the documentation for integrating Fivetran and dbt with Databricks and re-use the marketing_ analytics_demo project example to quickly get started.

The dbt task type in Databricks Workflows is in private preview. To try the dbt task type, please reach out to your Databricks account executive.

databricks

SECTION 5.2

Claims Automation on Databricks

Smart claims increases efficiency by automating all aspects of claims processing from ingestion, analysis, and decision–making.

by Anindita Mahapatra and Marzi Rasooli

According to the latest reports from global consultancy EY, the future of insurance will become increasingly **data–driven**, and **analytics enabled**. The recent focus on the cloud has improved access to advanced technological infrastructure, but most organizations still need help implementing and leveraging these capabilities. It's time to shift the focus on operationalizing services to realize value.

In today's economic circumstances, insurance companies face an ever-increasing number of challenges. Insurers are being forced to leverage data to their advantage and **innovate** at an accelerated pace. For personal P&C insurers, this means an increased focus on **personalization** and customer retention. Brand loyalty is at an all–time low, with customers continuously shopping for more competitive rates and better overall experiences, which increases the risk of churn. An increase in fraudulent claims further erodes profit margins. Insurers need to find additional ways to reduce costs and better manage risks.

Automating and optimizing the claims–handling process is one area that can significantly reduce costs through time saved and lesser reliance on human capital. Furthermore, effectively leveraging insights from data and advanced analytics can substantially reduce the overall exposure to risk.

The motivation behind the "Smart Claims" solution accelerator is simple — improve the claims handling process to enable faster settlement, lower processing costs, and deliver quicker insights about potentially fraudulent claims, with the lakehouse. Implementing the lakehouse paradigm simplifies the current architectural landscape and sets the scene for future expansion across the organization. The accompanying assets can be found here.

databricks

Reference Architecture and Workflow

A typical claims workflow involves some level of orchestration between operational systems such as Guidewire and analytical systems like Databricks. The diagram below shows an example of such a workflow for an automotive insurer.
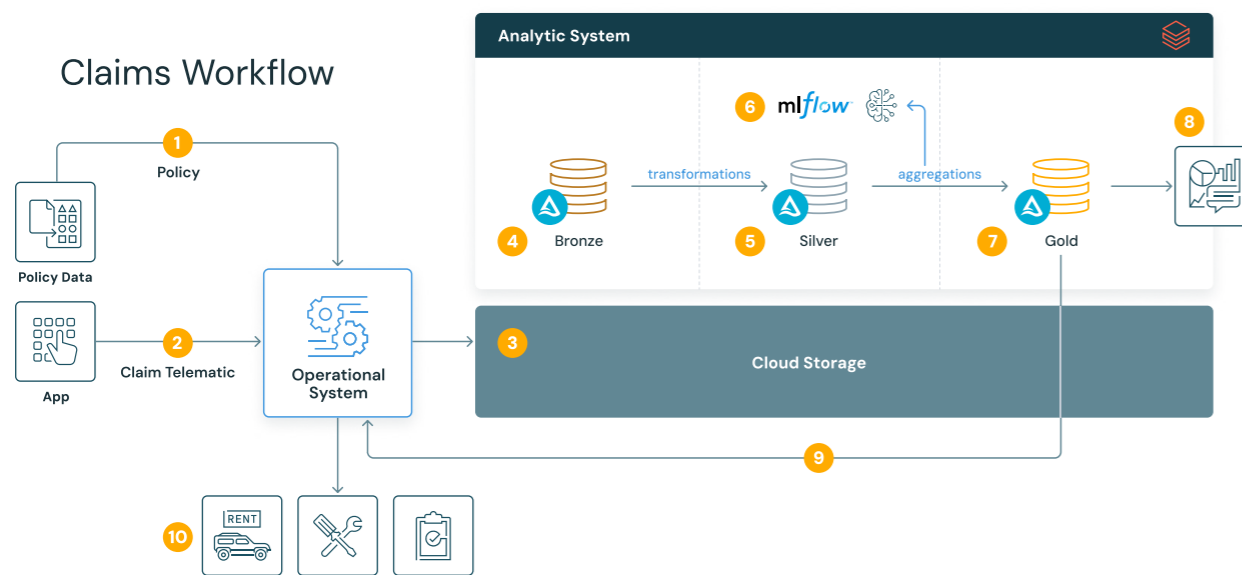
Smart Claims Reference Architecture and Workflow

Automating and optimizing the claims handling process requires a deep understanding of customer interaction with operational systems and the various sources of information available for analysis.

In this example, we assume that customers primarily interact through a mobile application, and from there, they can submit claims and monitor the status of existing cases. This touch point offers vital information about customer behavior. Another important source of information is IoT devices installed in customer vehicles. Telematics data can be streamed to the operational and analytical systems, providing valuable insights into customer-driving behavior and patterns. Other external data sources may include weather and road conditions data that supplement the traditional data categories such as vehicle characteristics (make, model, year), driver characteristics and exposure/coverage (limits, deductibles)

Access to additional data sources can become increasingly important, especially in the absence of data from traditional sources such as credit bureaus. Credit scores from bureaus usually form the basis for risk modeling, assessing the exposure for drivers, which ultimately impacts their premiums. On the other hand, data from mobile applications and IoT devices provide a more personalized view of customer behavior, which could be used to create a more accurate indicator of the risk associated with a given party. This alternative, **behavioral-based** approach to risk modeling and pricing is essential for delivering a hyper-personalized customer experience.

The lakehouse architecture powered by the Databricks Data Intelligence Platform is the only platform that combines all the required features and services to support a future-proof claims-handling process. From streaming to machine learning and reporting, Databricks offers the best platform to build an end-to-end solution for the insurance industry of tomorrow.

databricks

The following steps capture the overall flow:

- Policy data is ingested.
- Telematics data is continuously ingested from IoT sensors. A claimant submits claims data via a mobile app.
- All the operational data is ingested into cloud storage.
- This is incrementally loaded as 'Raw data' into Delta Bronze tables
- The data is wrangled and refined via various data transformations
- Data is scored using the trained model
- The predictions are loaded to a Gold table
- The Claims Dashboard is refreshed for visualization
- The resulting insights are fed back to the operational system. This provides the feedback loop of pulling data from Guidewire and passing 'Next Best Action' back to Guidewire in real time to know which claims should be prioritized.
- The Claims Decisioning workflows use these generated insights to route the case appropriately. (Eg. approve repair expenses, rental reimbursement, or alert authorities)

How the lakehouse paradigm aids Smart Claims

The lakehouse architecture enables all data personas (data engineers, data scientists, analytic engineers, and BI analysts) to work collaboratively on a single platform. Supporting all big-data workloads and paradigms (e.g., batch processing, streaming, DataOps, ML, MLOps, and BI) in a single, collaborative platform greatly simplifies the overall architecture, improves stability, and reduces cost significantly.

Databricks Delta Live Tables (DLT) pipelines offer a simple, declarative framework to develop and implement workloads quickly. It also provides native support for data quality management with granular constraints to guarantee the integrity of outputs.

ML and AI workloads can easily be created and managed with MLflow for reproducibility and auditability. MLFlow simplifies the entire model lifecycle, from experimenting through model deployment, serving, and archiving. ML can be run on all types of data including unstructured data beyond text (images, audio, video, etc). In this solution, we will use computer vision capabilities to assess damage to the vehicle.

Finally, Databricks SQL provides a fast and efficient engine to query curated and aggregated data. These insights can then be packaged and served through interactive Dashboards within minutes.

Unity Catalog provides a multi-cloud, centralized governance solution for all data and AI assets including files, tables, machine learning models and dashboards with built-in search, discovery, automated workload lineage.

databricks

The diagram below shows a reference architecture for the lakehouse in the context of typical insurance use cases:
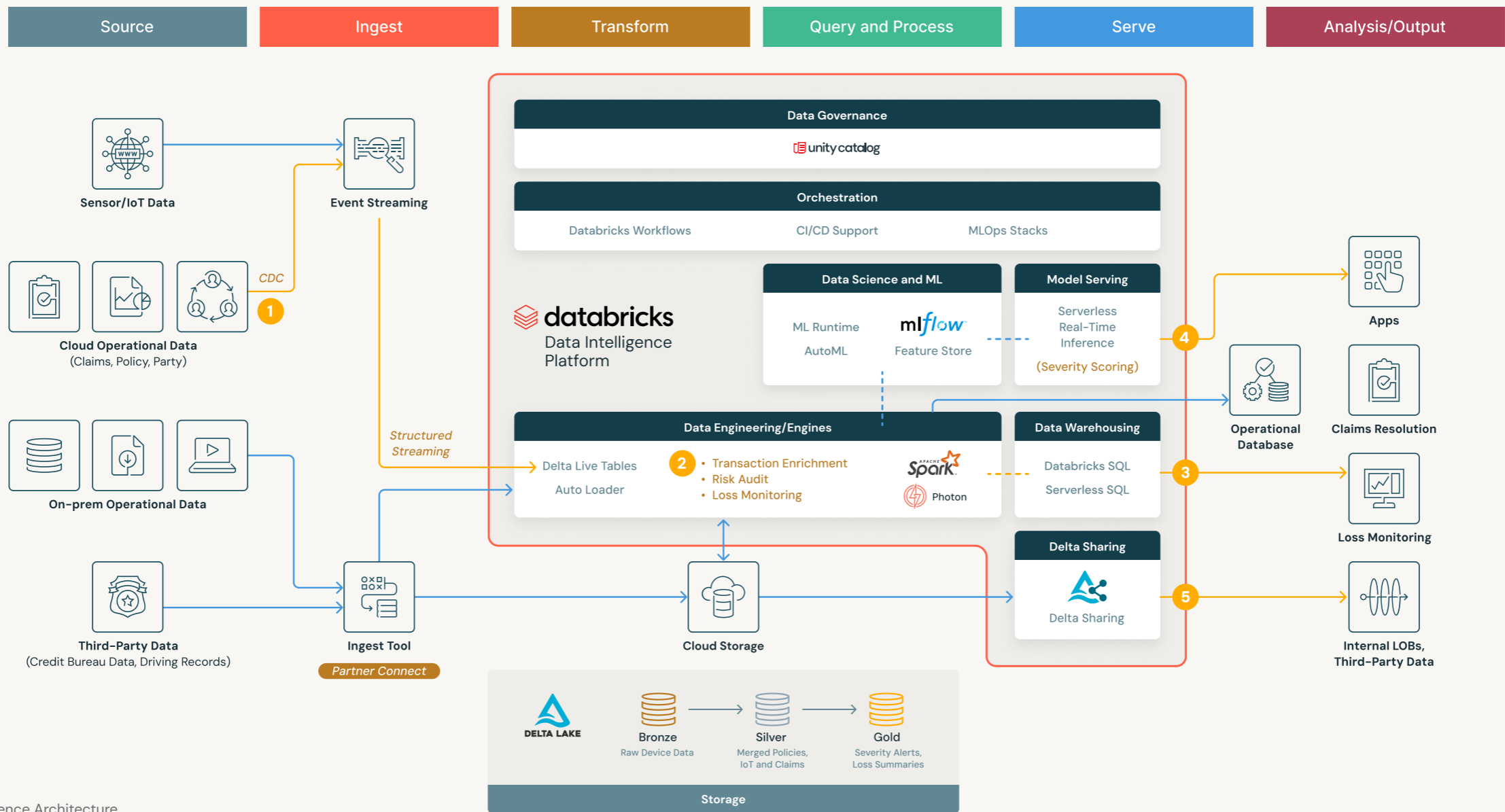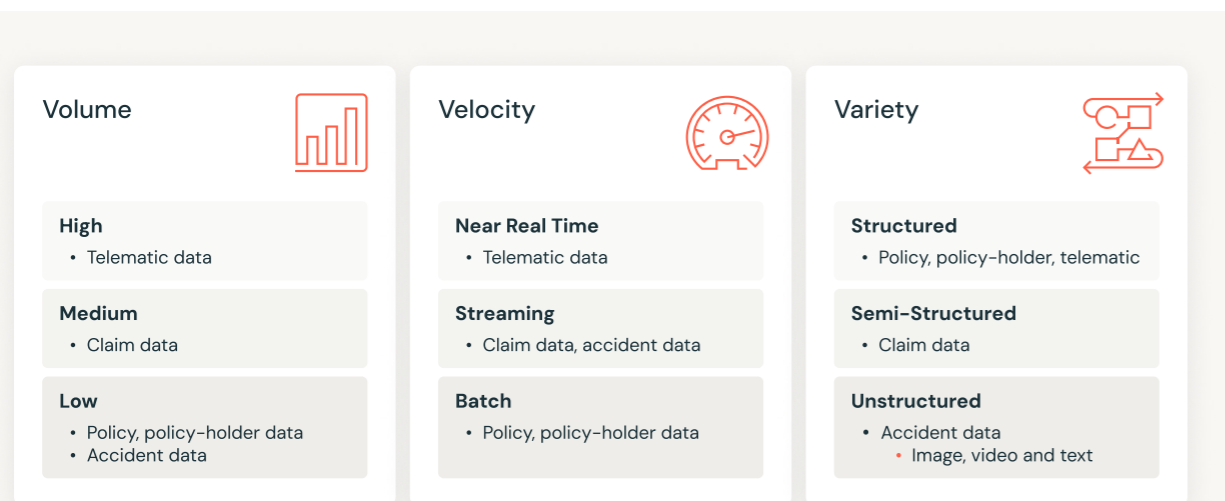

Insurance Reference Architecture

Data Ingestion using DLT and Multitask Workflows

Automating the claims-handling process starts with optimizing the ingestion and data engineering workflow. The figure below offers a summary of the typical data sources encountered including structured, semi-structured and unstructured. Some sources are slower-moving, while others update more rapidly. Additionally, some sources might be additive, requiring appending, while others offer incremental updates and must be treated as slow-changing dimensions.

Volume	Velocity	Variety
High • Telematic data	**Near Real Time** • Telematic data	**Structured** • Policy, policy-holder, telematic
Medium • Claim data	**Streaming** • Claim data, accident data	**Semi-Structured** • Claim data
Low • Policy, policy-holder data • Accident data	**Batch** • Policy, policy-holder data	**Unstructured** • Accident data • Image, video and text

Sample Datasets used in Claims Processing

DLT can simplify and operationalize the data processing pipeline. The framework offers support for Auto Loader to facilitate ingestion from streaming sources, efficient auto-scaling to handle sudden changes in data volumes, and resiliency via a restart of task failure.

Databricks Workflows can accommodate multiple tasks and workloads (e.g., notebooks, DLT, ML, SQL). Workflows support repair-and-run and compute sharing across tasks – enabling robust, scalable, cost-effective workloads. Additionally, Workflow can easily be automated through schedules or programmatic invoking via REST APIs.

Insight Generation using ML and Dynamic Rules Engine

Leveraging ML is essential to uncovering previously unknown patterns, highlighting new insights, and flagging suspicious activity. However, combining ML and traditional rules-based approaches can be even more powerful.

Within the claims-handling process, ML can be used for several use cases. One example would be using computer vision and ML for assessing and scoring images submitted with vehicle insurance claims. Models can be trained to focus on the validity and severity of damages. Here, MLFlow can be crucial in simplifying the model training and serving process with its end-to-end MLOps capabilities. MLFlow offers a serverless model serving through REST APIs. Trained models can be operationalized and put into production with the click of a button.

databricks

On the other hand, rules engines offer flexible ways of defining known operational characteristics and statistical checks, which can be automated and applied without requiring human interaction. Flags are raised whenever data does not comply with preset expectations and are sent for human review and investigation. Incorporating such an approach with ML-based workflows offers additional oversight and significantly reduces the time claims investigators require to dissect and review flagged cases.
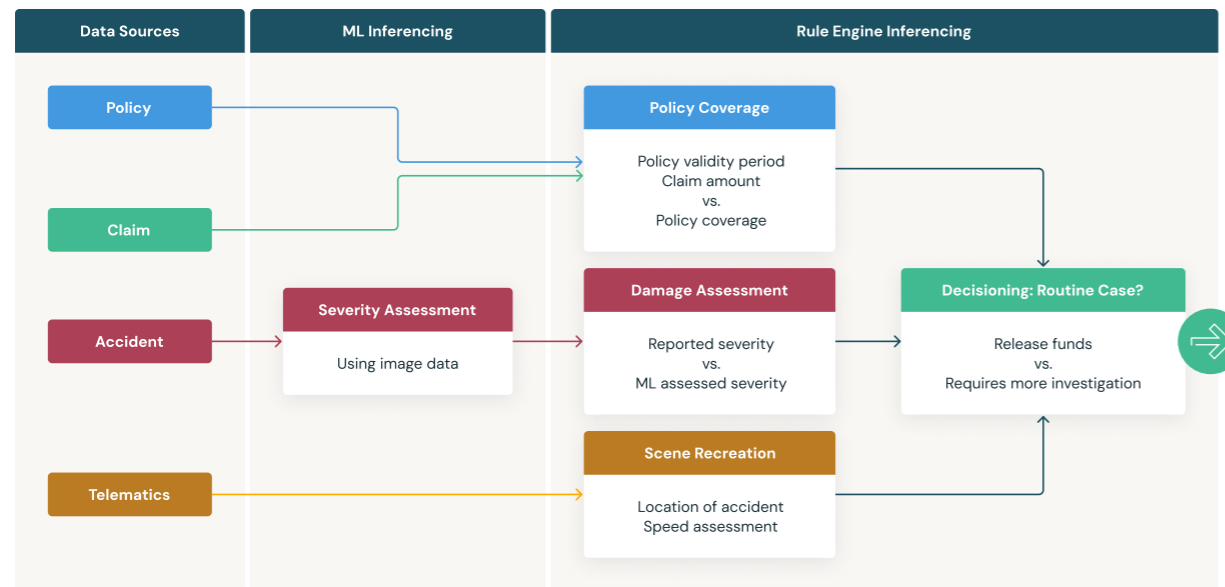

ML and Rule Engine Inferencing

Insight visualization using Dashboards

In this example, we created two dashboards to capture critical business insights. The dashboards include the following:

- A **Loss Summary** dashboard for a high-level view of the overall business operations; and

- A **Claims Investigation** dashboard with a granular view of claims details to understand the specifics of a given case.
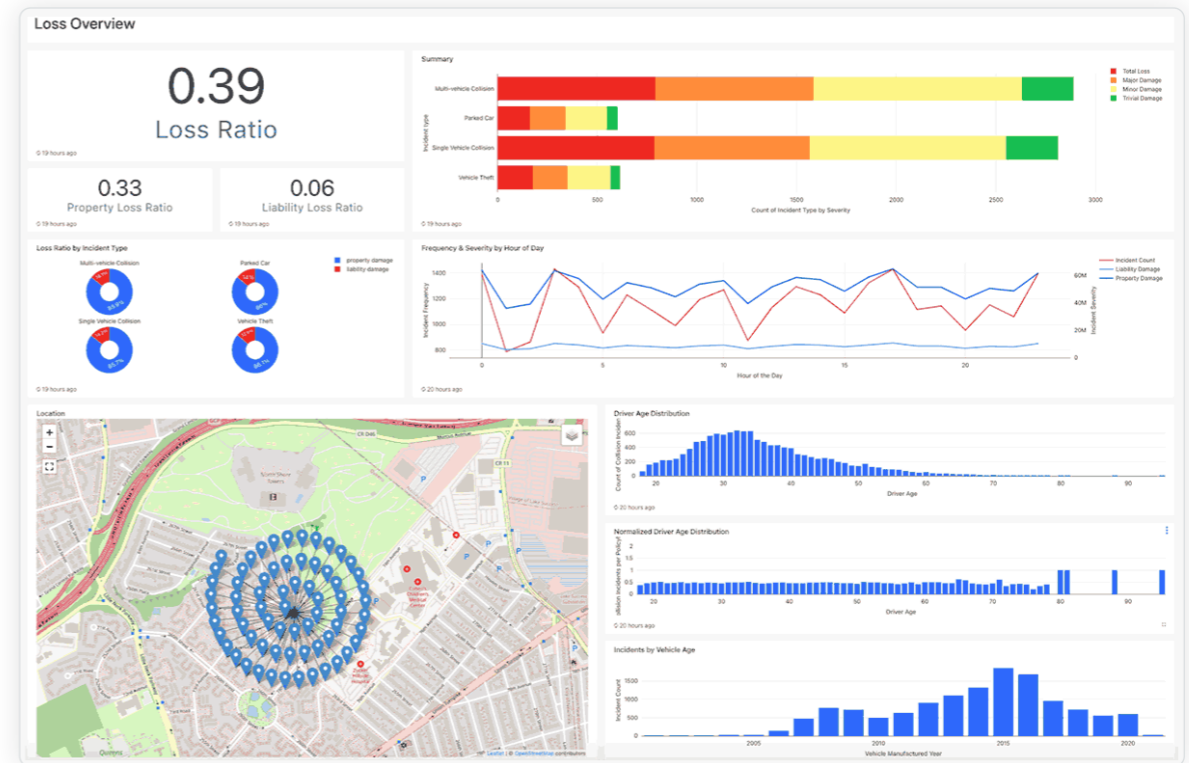
Analyzing recent trends can further aid in reviewing similar cases such as :

- Loss Ratio is computed by insurance claims paid plus adjustment expenses divided by total earned premiums. E.g. typical average Loss Ratio (all coverages combined, Bodily Injury, and Physical Damage) for personal auto should be around 65%

- Summary visualization captures count of incident type by damage severity

- Trend lines over various features/dimensions

- Geographic distribution of policies

The Claims Investigation dashboard facilitates faster investigation by providing all relevant information around a claim allowing the human investigator to drill down to a specific claim to see details such as images of the damaged vehicle, Claim, Policy and Driver details, Telematic data draws the path taken by the vehicle, Reported data is contrasted with assessed data insights.
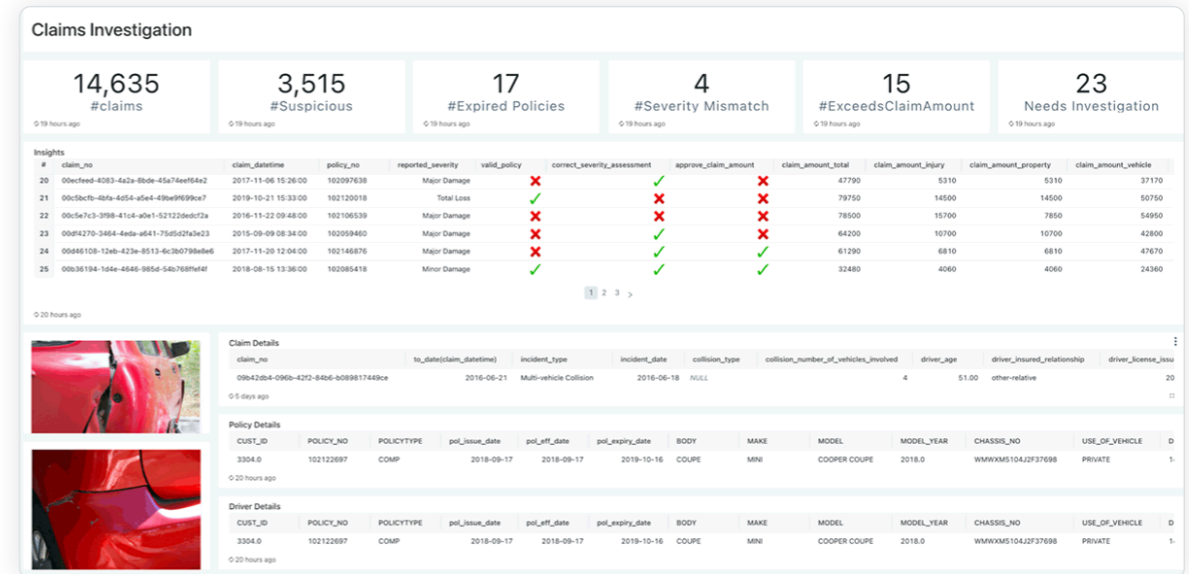
Claim Investigation Dashboard

Provides recent claims that are auto-scored in the pipeline using ML inferencing and rule engine

- A green tick is used to denote auto-assessment matches claims description

- A red cross indicates a mismatch that warrants further manual investigation

databricks

Summary

Innovation and personalization are essential for insurance firms to differentiate themselves from the competition. Databricks provides a data intelligence platform for insurers to enable and accelerate innovation with an open, secure, extensible architecture that easily integrates with third–party tools and services. This Solution Accelerator demonstrates how the paradigm can be applied to claims handling. Further, the Databricks ecosystem offers a range of capabilities to enable data teams and business stakeholders to collaborate and generate and share insights that support business decisions and drive tangible value to the bottom line.

The technical assets, including pipeline configurations, models, and sample data used in this example, can be accessed here or directly on Git.

databricks

SECTION 5.3

Design Patterns for Batch Processing in Financial Services

Laying the foundation for automating workflows

by Eon Retief

Introduction

Financial services institutions (FSIs) around the world are facing unprecedented challenges ranging from market volatility and political uncertainty to changing legislation and regulations. Businesses are forced to accelerate digital transformation programs; automating critical processes to reduce operating costs and improve response times. However, with data typically scattered across multiple systems, accessing the information required to execute on these initiatives tends to be easier said than done.

Architecting an ecosystem of services able to support the plethora of data-driven use cases in this digitally transformed business can, however, seem to be an impossible task. This blog will focus on one crucial aspect of the modern data stack: batch processing. A seemingly outdated paradigm, we'll see why batch processing remains a vital and highly viable component of the data architecture. And we'll see how Databricks can help FSIs navigate some of the crucial challenges faced when building infrastructure to support these scheduled or periodic workflows.

Why batch ingestion matters

Over the last two decades, the global shift towards an instant society has forced organizations to rethink the operating and engagement model. A digital-first strategy is no longer optional but vital for survival. Customer needs and demands are changing and evolving faster than ever. This desire for instant gratification is driving an increased focus on building capabilities that support real-time processing and decisioning. One might ask whether batch processing is still relevant in this new dynamic world.

While real-time systems and streaming services can help FSIs remain agile in addressing the volatile market conditions at the edge, they do not typically meet the requirements of back-office functions. Most business decisions are not reactive but rather, require considered, strategic reasoning. By definition, this approach requires a systematic review of aggregate data collected over a period of time. Batch processing in this context still provides the most efficient and cost-effective method for processing large, aggregate volumes of data. Additionally, batch processing can be done offline, reducing operating costs and providing greater control over the end-to-end process.

The world of finance is changing, but across the board incumbents and startups continue to rely heavily on batch processing to power core business functions. Whether for reporting and risk management or anomaly detection and surveillance, FSIs require batch processing to reduce human error, increase the speed of delivery, and reduce operating costs.

databricks

Getting started

Starting with a 30,000-ft view, most FSIs will have a multitude of data sources scattered across on-premises systems, cloud-based services and even third-party applications. Building a batch ingestion framework that caters for all these connections require complex engineering and can quickly become a burden on maintenance teams. And that's even before considering things like change data capture (CDC), scheduling, and schema evolution. In this section, we will demonstrate how the lakehouse architecture for financial services and its ecosystem of partners can be leveraged to address these key challenges and greatly simplify the overall architecture.

The lakehouse architecture is designed to provide a unified platform that supports all analytical and scientific data workloads. Figure 1 shows the reference architecture for a decoupled design that allows easy integration with other platforms that support the modern data ecosystem. The lakehouse makes it easy to construct ingestion and serving layers that operate irrespective of the data's source, volume, velocity, and destination.
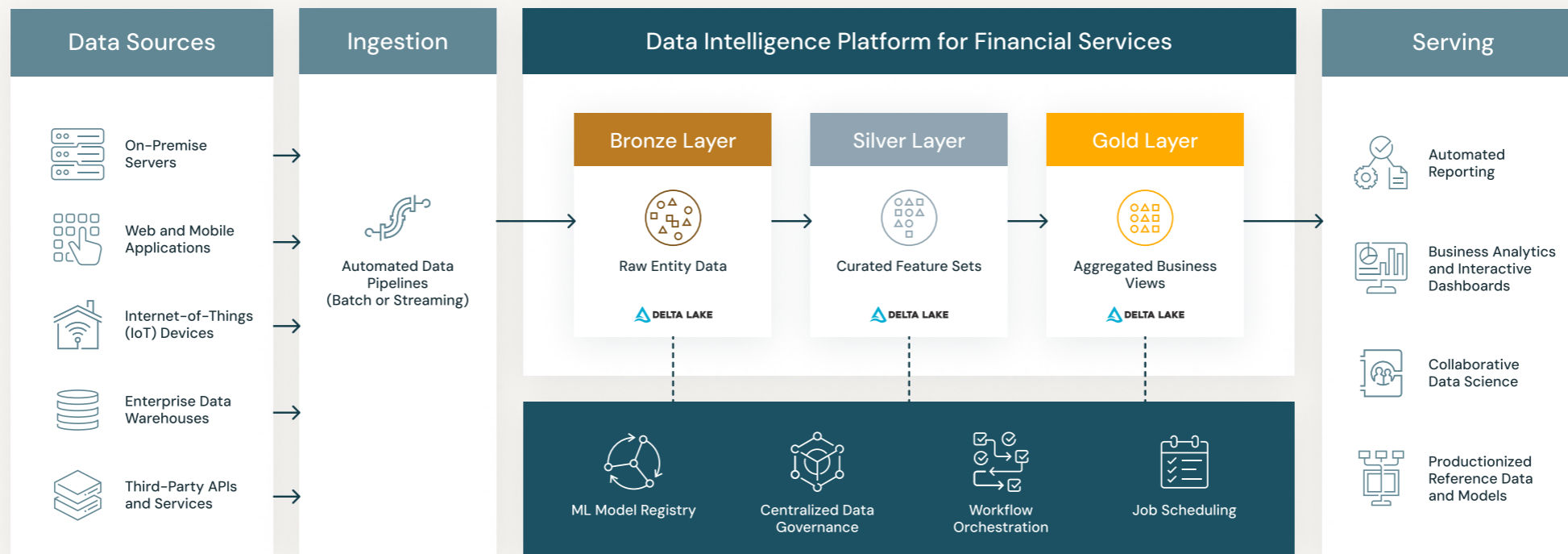

Figure 1 — Reference architecture of the lakehouse for financial services.

databricks

To demonstrate the power and efficiency of the LFS, we turn to the world of insurance. We consider the basic reporting requirements for a typical claims workflow. In this scenario, the organization might be interested in the key metrics driven by claims processes. For example:

- Number of active policies
- Number of claims
- Value of claims
- Total exposure
- Loss ratio

Additionally, the business might want a view of potentially suspicious claims and a breakdown by incident type and severity. All these metrics are easily calculable from two key sources of data: 1) the book of policies and 2) claims filed by customers. The policy and claims records are typically stored in a combination of enterprise data warehouses (EDWs) and operational databases. The main challenge becomes connecting to these sources and ingesting data into our lakehouse, where we can leverage the power of Databricks to calculate the desired outputs.

Luckily, the flexible design of the LFS makes it easy to leverage best-in-class products from a range of SaaS technologies and tools to handle specific tasks. One possible solution for our claims analytics use case would be to use Fivetran for the batch ingestion plane. Fivetran provides a simple and secure platform for connecting to numerous data sources and delivering data directly to the Databricks Data Intelligence Platform. Additionally, it offers native support for CDC, schema evolution and workload scheduling. In Figure 2, we show the technical architecture of a practical solution for this use case.
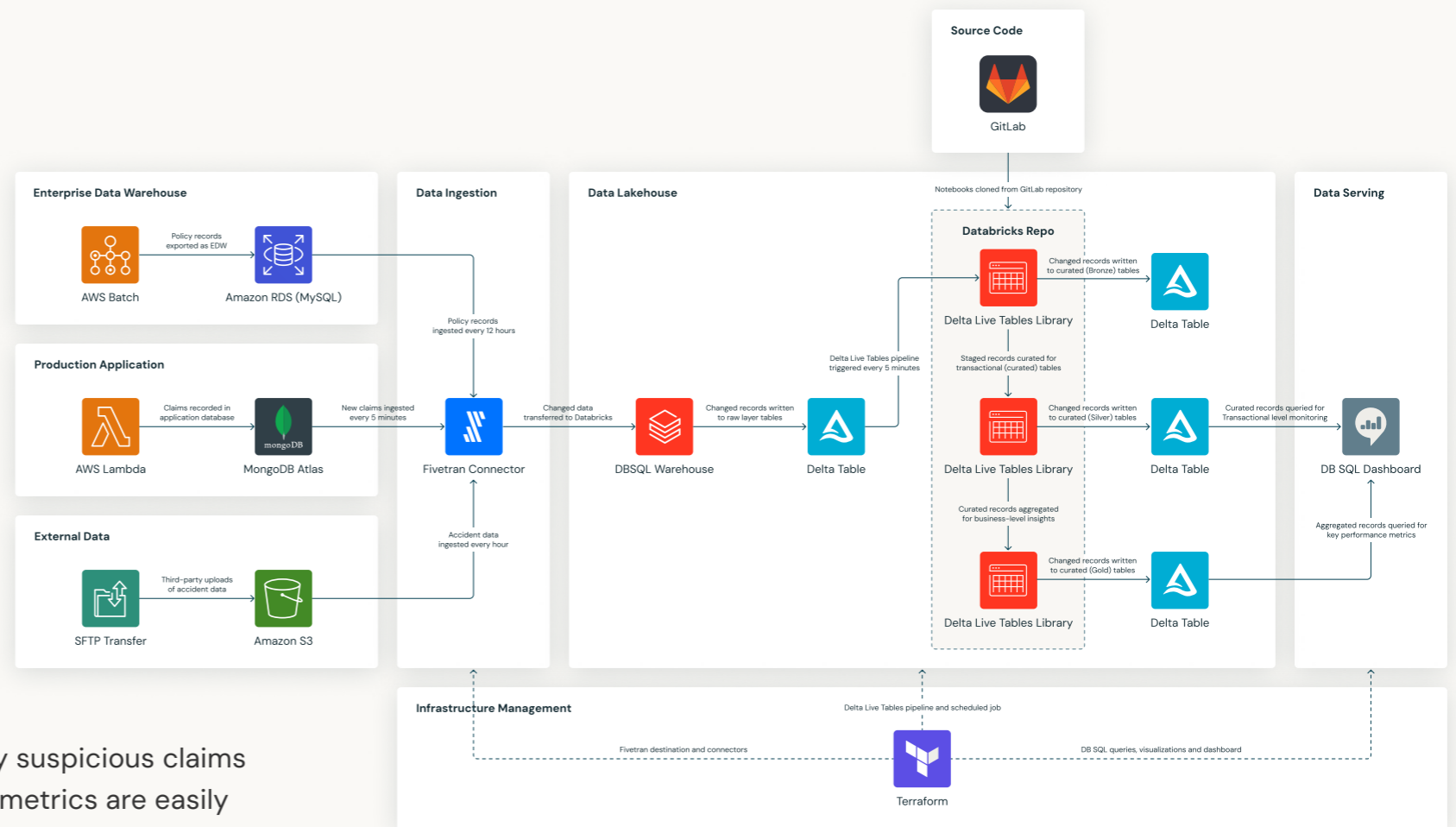

Figure 2 — Technical architecture for a simple insurance claims workflow.

Once the data is ingested and delivered to the LFS, we can use Delta Live Tables (DLT) for the entire engineering workflow. DLT provides a simple, scalable declarative framework for automating complex workflows and enforcing data quality controls. The outputs from our DLT workflow, our curated and aggregated assets, can be interrogated using Databricks SQL (DB SQL). DB SQL brings data warehousing to the LFS to power business-critical analytical workloads. Results from DB SQL queries can be packaged in easy-to-consume dashboards and served to business users.

Step 1: Creating the ingestion layer

Setting up an ingestion layer with Fivetran requires a two-step process. First, configuring a so-called destination where data will be delivered, and second, establishing one or more connections with the source systems. The Partner Connect interface takes care of the first step with a simple, guided interface to connect Fivetran with a Databricks warehouse. Fivetran will use the warehouse to convert raw source data to Delta tables and store the results in the Databricks Data Intelligence Platform. Figures 3 and 4 show steps from the Partner Connect and Fivetran interfaces to configure a new destination.

Figure 3 — Databricks Partner Connect interface for creating a new connection.

Figure 4 — Fivetran interface for confirming a new destination.

databricks

For the next step, we move to the Fivetran interface. From here, we can easily create and configure connections to several different source systems (please refer to the official documentation for a complete list of all supported connections). In our example, we consider three sources of data: 1) policy records stored in an Operational Data Store (ODS) or Enterprise Data Warehosue (EDW), 2) claims records stored in an operational database, and 3) external data delivered to blob storage. As such, we require three different connections to be configured in Fivetran. For each of these, we can follow Fivetran's simple guided process to set up a connection with the source system. Figures 5 and 6 show how to configure new connections to data sources.
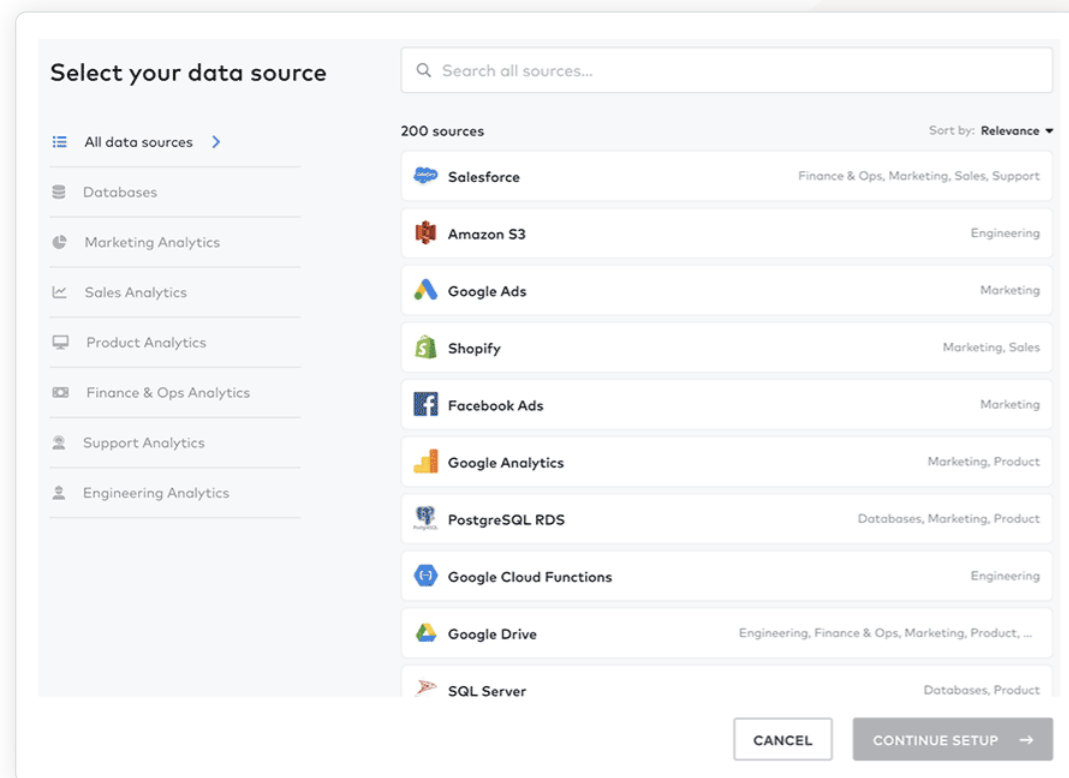

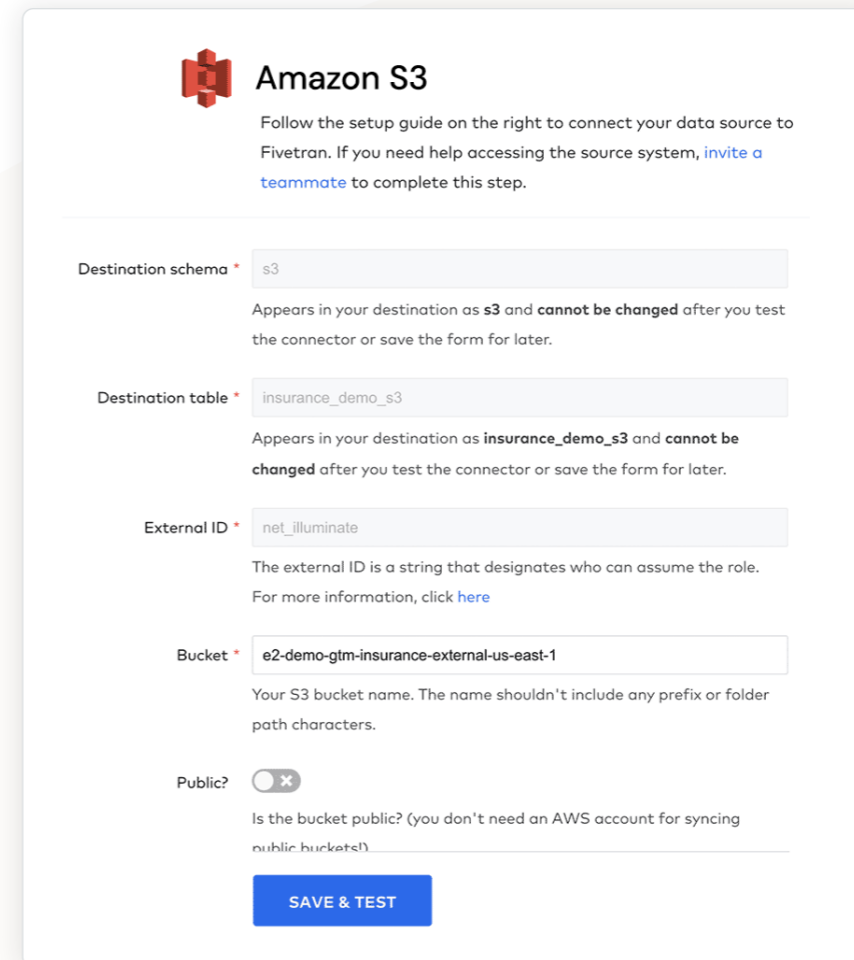Figure 6 — Fivetran interface for configuring a data source connection.

Figure 5 — Fivetran interface for selecting a data source type.

databricks

Connections can further be configured once they have been validated. One important option to set is the frequency with which Fivetran will interrogate the source system for new data. In Figure 7, we can see how easy Fivetran has made it to set the sync frequency with intervals ranging from 5 minutes to 24 hours.
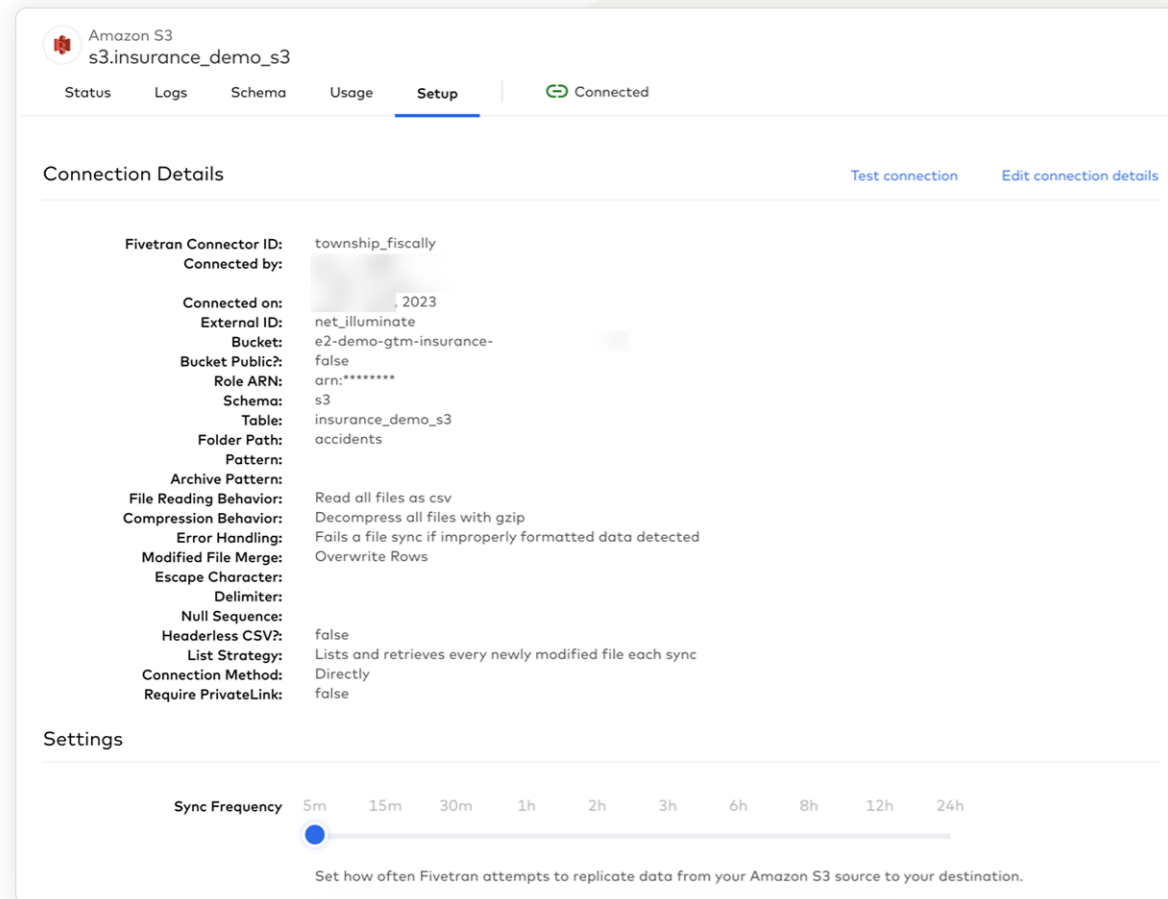

Figure 7 — Overview of configuration for a Fivetran connector.

Fivetran will immediately interrogate and ingest data from source systems once a connection is validated. Data is stored as Delta tables and can be viewed from within Databricks through the Catalog Explorer. By default, Fivetran will store all data under the Hive metastore. A new schema is created for each new connection, and each schema will contain at least two tables: one containing the data and another with logs from each attempted ingestion cycle (see Figure 8).
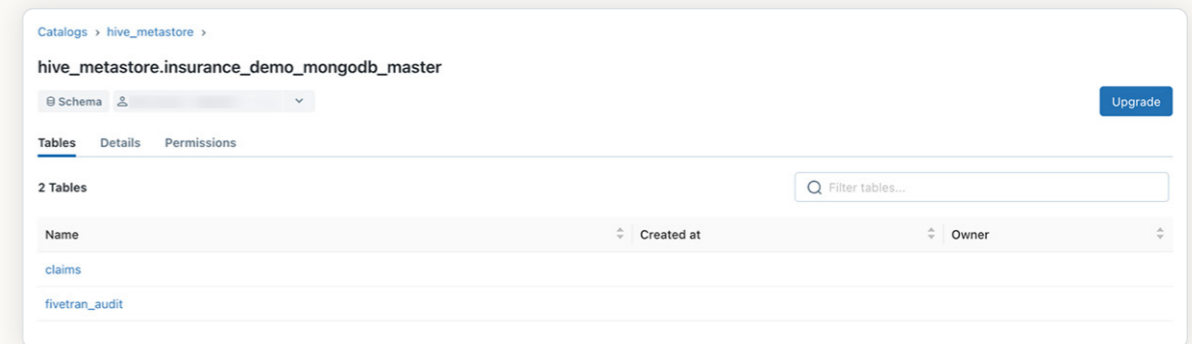
Figure 8 — Summary of tables created by Fivetran in the Databricks Warehouse for an example connection.

Having the data stored in Delta tables is a significant advantage. Delta Lake natively supports granular data versioning, meaning we can time travel through each ingestion cycle (see Figure 9). We can use DB SQL to interrogate specific versions of the data to analyze how the source records evolved.

Figure 9 — View of the history showing changes made to the Fivetran audit table.

It is important to note that if the source data contains semi-structured or unstructured values, those attributes will be flattened during the conversion process. This means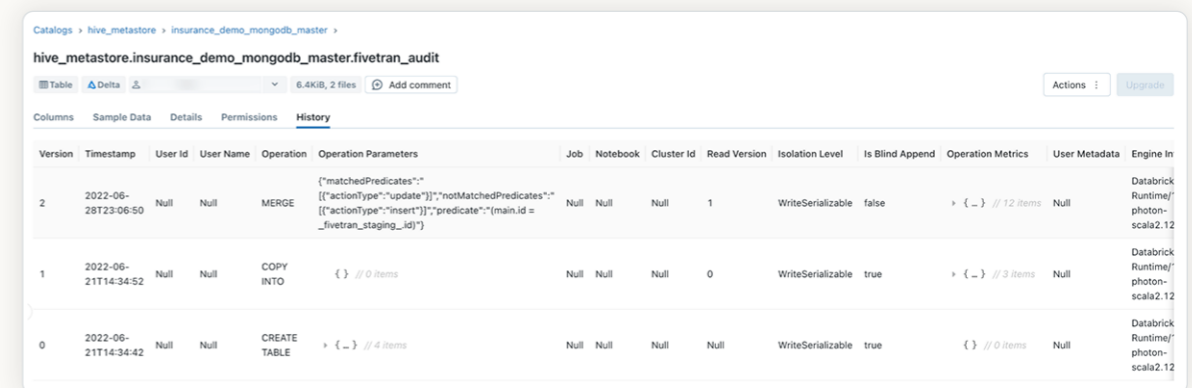 that the results will be stored in grouped text-type columns, and these entities will have to be dissected and unpacked with DLT in the curation process to create separate attributes.

Step 2: Automating the workflow

With the data in the lakehouse, we can use Delta Live Tables (DLT) to build a simple, automated data engineering workflow. DLT provides a declarative framework for specifying detailed feature engineering steps. Currently, DLT supports APIs for both Python and SQL. In this example, we will use Python APIs to build our workflow.

The most fundamental construct in DLT is the definition of a table. DLT interrogates all table definitions to create a comprehensive workflow for how data should be processed. For instance, in Python, tables are created using function definitions and the `dlt.table` decorator (see example of Python code below). The decorator is used to specify the name of the resulting table, a descriptive comment explaining the purpose of the table, and a collection of table properties.

```python
@dlt.table(
    name            = "curated_claims",
    comment         = "Curated claim records",
    table_properties = {
        "layer": "silver",
        "pipelines.autoOptimize.managed": "true",
        "delta.autoOptimize.optimizeWrite": "true",
        "delta.autoOptimize.autoCompact": "true"
    }
)
def curate_claims():
    # Read the staged claim records into memory
    staged_claims = dlt.read("staged_claims")
    # Unpack all nested attributes to create a flattened table structure
    curated_claims = unpack_nested(df = staged_claims, schema = schema_claims)

    ...
```

Instructions for feature engineering are defined inside the function body using standard PySpark APIs and native Python commands. The following example shows how PySpark joins claims records with data from the policies table to create a single, curated view of claims.

databricks

```
1       ···
2
3           # Read the staged claim records into memory
4           curated_policies = dlt.read("curated_policies")
5           # Evaluate the validity of the claim
6           curated_claims = curated_claims \
7               .alias("a") \
8               .join(
9                   curated_policies.alias("b"),
10                  on  = F.col("a.policy_number") == F.col("b.policy_number"),
12                  how = "left"
13              ) \
14              .select([F.col(f"a.{c}") for c in curated_claims.columns] + [F.col(f"b.
15      {c}").alias(f"policy_{c}") for c in ("effective_date", "expiry_date")]) \
16              .withColumn(
17                  # Calculate the number of months between coverage starting and the
18      claim being filed
19                  "months_since_covered", F.round(F.months_between(F.col("claim_date"),
20      F.col("policy_effective_date")))
21              ) \
22              .withColumn(
23                  # Check if the claim was filed before the policy came into effect
24                  "claim_before_covered", F.when(F.col("claim_date") < F.col("policy_
25      effective_date"), F.lit(1)).otherwise(F.lit(0))
26              ) \
27              .withColumn(
28                  # Calculate the number of days between the incident occurring and the
29      claim being filed
30                  "days_between_incident_and_claim", F.datediff(F.col("claim_date"),
31      F.col("incident_date"))
32              )
33
34          # Return the curated dataset
35          return curated_claims
```

One significant advantage of DLT is the ability to specify and enforce data quality standards. We can set expectations for each DLT table with detailed data quality constraints that should be applied to the contents of the table. Currently, DLT supports expectations for three different scenarios:

Decorator	Description
expect	Retain records that violate expectations
expect_or_drop	Drop records that violate expectations
expect_or_fail	Halt the execution if any record(s) violate constraints

Expectations can be defined with one or more data quality constraints. Each constraint requires a description and a Python or SQL expression to evaluate. Multiple constraints can be defined using the expect_all, expect_all_or_drop, and expect_all_or_fail decorators. Each decorator expects a Python dictionary where the keys are the constraint descriptions, and the values are the respective expressions. The example below shows multiple data quality constraints for the retain and drop scenarios described above.

```
1       @dlt.expect_all({
2           "valid_driver_license": "driver_license_issue_date > (current_date() -
3       cast(cast(driver_age AS INT) AS INTERVAL YEAR))",
4           "valid_claim_amount": "total_claim_amount > 0",
5           "valid_coverage": "months_since_covered > 0",
6           "valid_incident_before_claim": "days_between_incident_and_claim > 0"
7       })
8       @dlt.expect_all_or_drop({
9           "valid_claim_number": "claim_number IS NOT NULL",
10          "valid_policy_number": "policy_number IS NOT NULL",
12          "valid_claim_date": "claim_date < current_date",
13          "valid_incident_date": "incident_date < current_date",
14          "valid_incident_hour": "incident_hour between 0 and 24",
15          "valid_driver_age": "driver_age > 16",
16          "valid_effective_date": "policy_effective_date < current_date()",
17          "valid_expiry_date": "policy_expiry_date <= current_date()"
18      })
19      def curate_claims():
20          ...
```

We can use more than one Databricks Notebook to declare our DLT tables. Assuming we follow the medallion architecture, we can, for example, use different notebooks to define tables comprising the Bronze, Silver and Gold layers. The DLT framework can digest instructions defined across multiple notebooks to create a single workflow; all inter-table dependencies and relationships are processed and considered automatically. Figure 10 shows the complete workflow for our claims example. Starting with three source tables, DLT builds a comprehensive pipeline that delivers thirteen tables for business consumption.
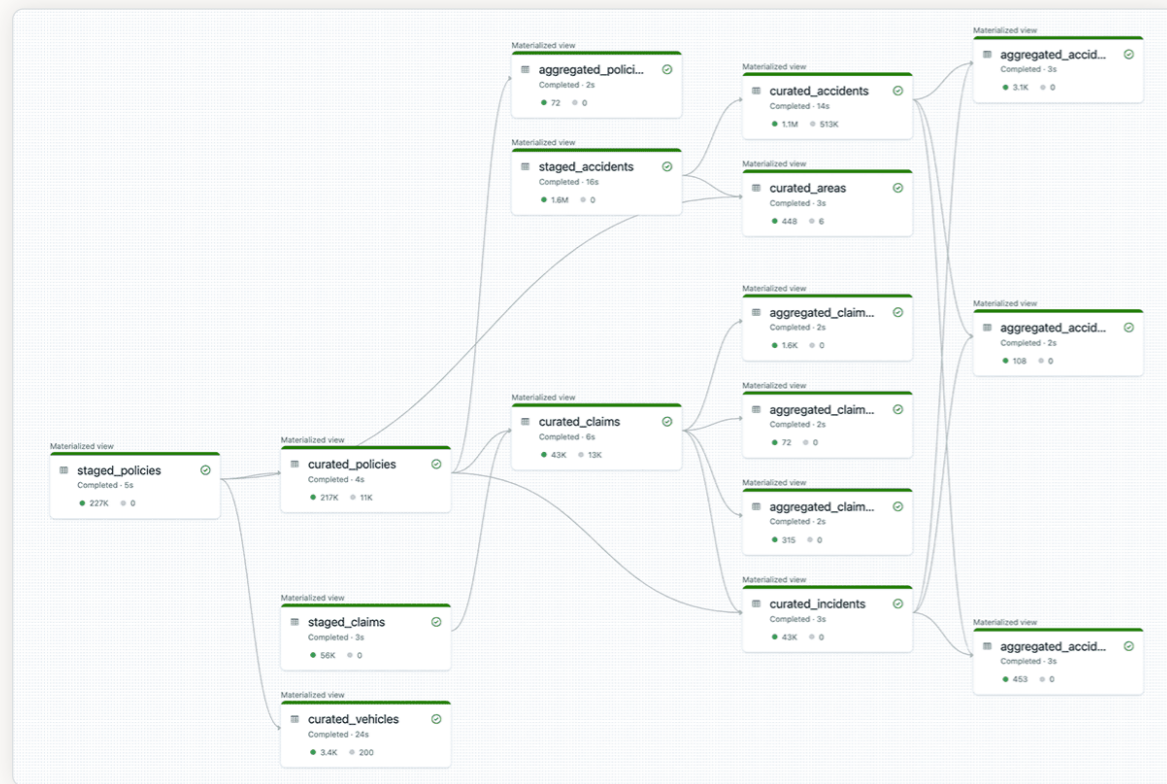

Figure 10 — Overview of a complete Delta Live Tables (DLT) workflow.

Results for each table can be inspected by selecting the desired entity. Figure 11 provides an example of the results of the curated claims table. DLT provides a high-level overview of the results from the data quality controls:
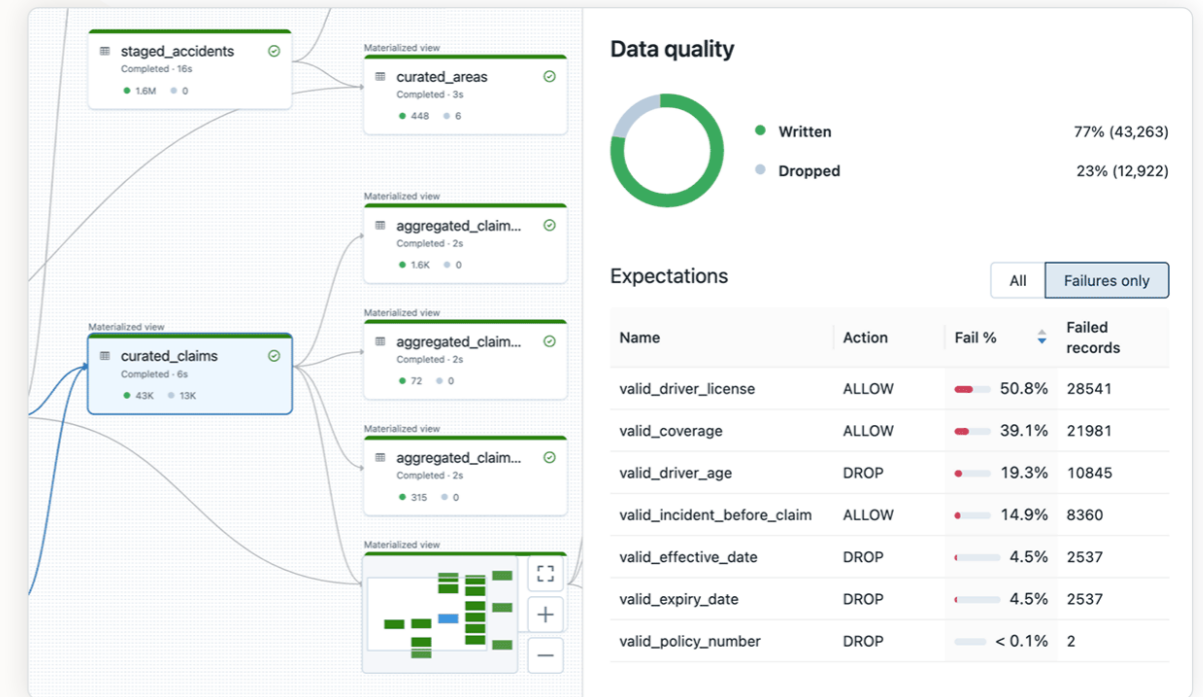

Figure 11 — Example of detailed view for a Delta Live Tables (DLT) table entity with the associated data quality report.

Results from the data quality expectations can be analyzed further by querying the event log. The event log contains detailed metrics about all expectations defined for the workflow pipeline. The query below provides an example for viewing key metrics from the last pipeline update, including the number of records that passed or failed expectations:

```
1    SELECT
2      row_expectations.dataset AS dataset,
3      row_expectations.name AS expectation,
4      SUM(row_expectations.passed_records) AS passing_records,
5      SUM(row_expectations.failed_records) AS failing_records
6    FROM
7      (
8        SELECT
9          explode(
10           from_json(
12             details :flow_progress :data_quality :expectations,
13             "array<struct<name: string, dataset: string, passed_records: int,
14   failed_records: int>>"
15           )
16         ) row_expectations
17       FROM
18         event_log_raw
19       WHERE
20         event_type = 'flow_progress'
21         AND origin.update_id = '${latest_update.id}'
22     )
23   GROUP BY
24     row_expectations.dataset,
25     row_expectations.name;
```

Again, we can view the complete history of changes made to each DLT table by looking at the Delta history logs (see Figure 12). It allows us to understand how tables evolve over time and investigate complete threads of updates if a pipeline fails.

Figure 12 — View the history of changes made to a resulting Delta Live Tables (DLT) table entity.

We can further use change data capture (CDC) to update tables based on changes in the source datasets. DLT CDC supports updating tables with slow-changing dimensions (SCD) types 1 and 2.

We have one of two options for our batch process to trigger the DLT pipeline. We can use the Databricks Auto Loader to incrementally process new data as it arrives in the source tables or create scheduled jobs that trigger at set times or intervals. In this example, we opted for the latter with a scheduled job that executes the DLT pipeline every five minutes.

databricks

Operationalizing the outputs

The ability to incrementally process data efficiently is only half of the equation. Results from the DLT workflow must be operationalized and delivered to business users. In our example, we can consume outputs from the DLT pipeline through ad hoc analytics or prepacked insights made available through an interactive dashboard.

Ad hoc analytics

Databricks SQL (or DB SQL) provides an efficient, cost-effective data warehouse on top of the lakehouse architecture. It allows us to run our SQL workloads directly against the source data with up to 12x better price/performance than its alternatives.

We can leverage DB SQL to perform specific ad hoc queries against our curated and aggregated tables. We might, for example, run a query against the curated policies table that calculates the total exposure. The DB SQL query editor provides a simple, easy-to-use interface to build and execute such queries (see example below).

```
1   SELECT
2     round(curr.total_exposure, 0) AS total_exposure,
3     round(prev.total_exposure, 0) AS previous_exposure
4   FROM
5     (
6       SELECT
7         sum(sum_insured) AS total_exposure
8       FROM
9         insurance_demo_lakehouse.curated_policies
10      WHERE
12        expiry_date > '{{ date.end }}'
13        AND (effective_date <= '{{ date.start }}'
14          OR (effective_date BETWEEN '{{ date.start }}' AND '{{ date.end }}'))
15    ) curr
16    JOIN
17    (
18      SELECT
19        ...
```

We can also use the DB SQL query editor to run queries against different versions of our Delta tables. For example, we can query a view of the aggregated claims records for a specific date and time (see example below). We can further use DB SQL to compare results from different versions to analyze only the changed records between those states.

```
1   SELECT
2     *
3   FROM
4     insurance_demo_lakehouse.aggregated_claims_weekly TIMESTAMP AS OF '2022-06-
5   05T17:00:00';
```

DB SQL offers the option to use a serverless compute engine, eliminating the need to configure, manage or scale cloud infrastructure while maintaining the lowest possible cost. It also integrates with alternative SQL workbenches (e.g., DataGrip), allowing analysts to use their favorite tools to explore the data and generate insights.

databricks

Business insights

Finally, we can use DB SQL queries to create rich visualizations on top of our query results. These visualizations can then be packaged and served to end users through interactive dashboards (see Figure 13).
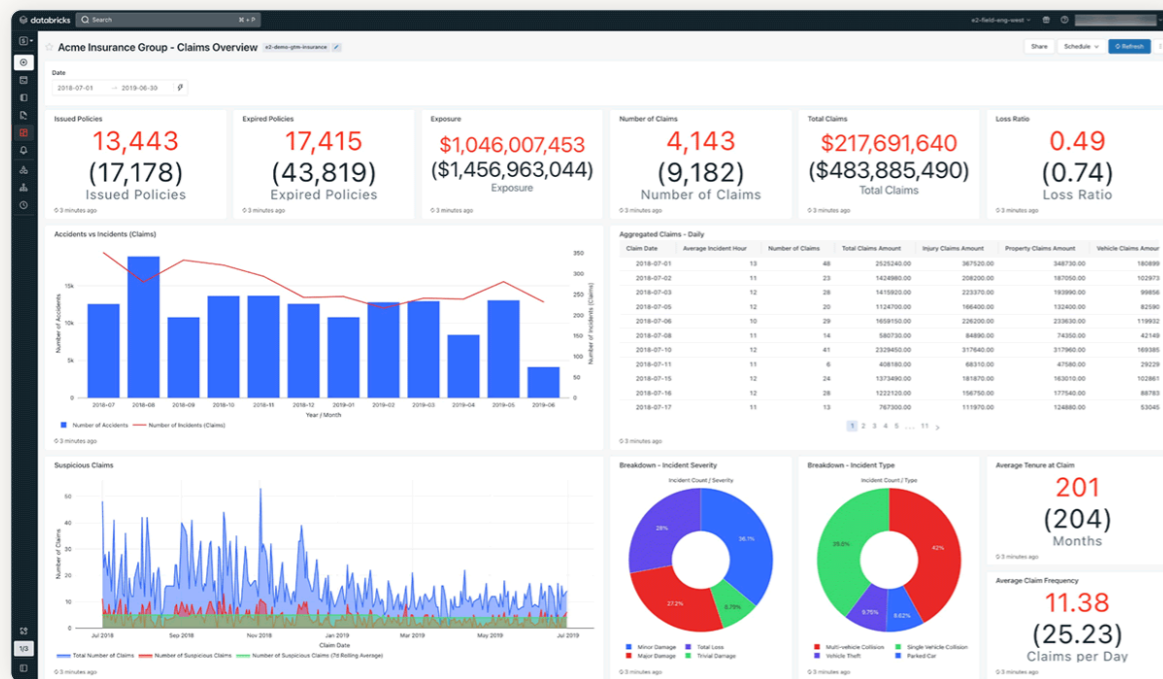

Figure 13 — Example operational dashboard built on a set of resulting Delta Live Tables (DLT) table entities.

For our use case, we created a dashboard with a collection of key metrics, rolling calculations, high-level breakdowns, and aggregate views. The dashboard provides a complete summary of our claims process at a glance. We also added the option to specify specific date ranges. DB SQL supports a range of query parameters that can substitute values into a query at runtime. These query parameters can be defined at the dashboard level to ensure all related queries are updated accordingly.

DB SQL integrates with numerous third-party analytical and BI tools like Power BI, Tableau and Looker. Like we did for Fivetran, we can use Partner Connect to link our external platform with DB SQL. This allows analysts to build and serve dashboards in the platforms that the business prefers without sacrificing the performance of DB SQL and the Databricks Data Intelligence Platform.

Conclusion

As we move into this fast-paced, volatile modern world of finance, batch processing remains a vital part of the modern data stack, able to hold its own against the features and benefits of streaming and real-time services. We've seen how we can use the lakehouse architecture for financial services and its ecosystem of partners to architect a simple, scalable, and extensible framework that supports complex batch-processing workloads with a practical example in insurance claims processing. With Delta Live Tables (DLT) and Databricks SQL (DB SQL), we can build a data platform with an architecture that scales infinitely, is easy to extend to address changing requirements, and will withstand the test of time.

To learn more about the sample pipeline described, including the infrastructure setup and configuration used, please refer to this GitHub repository or watch this demo video.

databricks

SECTION 5.4

Enabling Game Teams to Engage Directly With Their Data With AI/BI Genie Spaces

by Huntting Buckley and Duncan Davis

Introduction

When we announced the next evolution of Databricks, the Data Intelligence Platform, in November 2023, we shared plans to integrate machine learning, GenAI and other capabilities into our platform. With this we would further your productivity and the value you could generate from your data. Through this blog we'll explore AI/BI Genie, which is designed to democratize the next generation of data insight.

A Genie space provides your users the ability to engage with and explore the data itself by creating a chat interface to your data. Instead of being limited to what a dashboard might show them, users can ask their own questions and dive deeper into details. You point Genie at a series of tables and ask questions of the data. The Genie space interprets the question and presents results. With the space, users are able to explore their ideas as though they're working side-by-side with a data analyst. It can even turn the results into visualizations for the user. For a detailed view of Genie spaces, check out the documentation.

To demonstrate, we're using a public World of Warcraft (WoW) dataset to create a Genie space, re-create a user journey and explore the value it can bring to a games company. It's easy to set up a Genie space, and we encourage you to try it with one of your own datasets. It'll be more meaningful and as long as you're Unity Catalog–enabled, it'll be easier to set up than the sample space.

Creating a Genie space

As a reminder, for this to work, you need to be on Unity Catalog or have the datasets you're going to use in Unity Catalog. Genie spaces is still in Public Preview, so your first step will be to enable it from the workspace-level Previews Page. Once enabled you'll see a Genie button on the left side of your Databricks workspace. Click there and then click New in the top-right corner, give it a title and assign it a warehouse and add tables.

When creating a Genie space and selecting the tables to include, think about the end user. Who is going to be asking questions? What types of questions are they going to ask? What tables might support those questions? You may include Silver and/or Gold tables for the domain in question. Try to avoid datasets completely disconnected from each other as this will negatively impact your results.

For this discussion, we're using a Kaggle dataset that includes information about characters in WoW over a period of three years. The datasets include high-level session information, character-level progression, character class, zones they entered during a session and other related details like guild membership. As we explore this Genie space we're thinking about it through the lens of a game designer who's new to this capability and is interested in understanding character engagement across the open world.

databricks

Getting started with the Genie space

When you enter the space there's a button that will analyze the datasets and provide you with some example questions you could answer with data included in this space. By default these are autogenerated by the Genie space based on the datasets included, but you can add your own examples when creating the space. These prompts help a new user understand the types of things this space may be able to answer to help them get started. Your user can also ask the space, as we do in the next diagram, to describe the data it has access to. It'll generate details about the tables, their columns and what it might all mean.

Interrogating and iterating on your data

For this Genie space we're using a Kaggle dataset of WoW horde character data across a period of three years. This dataset's tables are well connected and we imported all tables. We then experimented with the Genie space as if we were a game designer brand new to the capability who's looking to learn more about the character experience across our game world.

To start our effort, we ask the space to tell us a bit about the tables themselves. We see that the space has information about character levels, different map zones, session information and details about the characters in the game, including level, race, class and which zone they were in at a point in time. It also tells us that the sessionIDs and the char column, which represents characters in the game, are the two core primary keys across these tables. To get used to the interface, we start with a simple question: "What's the most commonly race first played by a player?" It returns an answer, but not quite the answer we're asking for. Why is that? Because these tables don't have playerIDs, just CharacterIDs. The space returns the closest thing that makes sense to it: the most commonly played race based on the number of characters.


databricks

This example shows us a few important things

1. The model can handle questions even when not written well: "most commonly race" vs. "what is the most commonly picked race"

2. The model will attempt to answer the question and find something that seems right: character vs. player. This is both good and bad. The model won't know that something wasn't answered exactly right, so you've got to be paying attention.

3. In addition to providing an answer, the model provides an explanation of what it returned to help with #2. "This query retrieves the race with the highest number of characters . . . ." You'll notice that it doesn't actually say "player" in the explanation.

Just because Blood Elves are the most–created character doesn't mean that players are actually engaging with them. You might ask for a ranked list of all races by character count or by time played. We didn't dig deeper there, instead choosing to dive deeper into engagement with Blood Elves by asking: "What is the level distribution for blood elves?"

This table is useful, and if you're a numbers person you'll immediately see that the numbers are dropping quickly as you go from level 1 to level 10. For this type of analysis, a visualization might be easier to consume. After asking the space a question, there'll be a button labeled Visualize. When we click the Visualize button, the Genie space returns this chart.
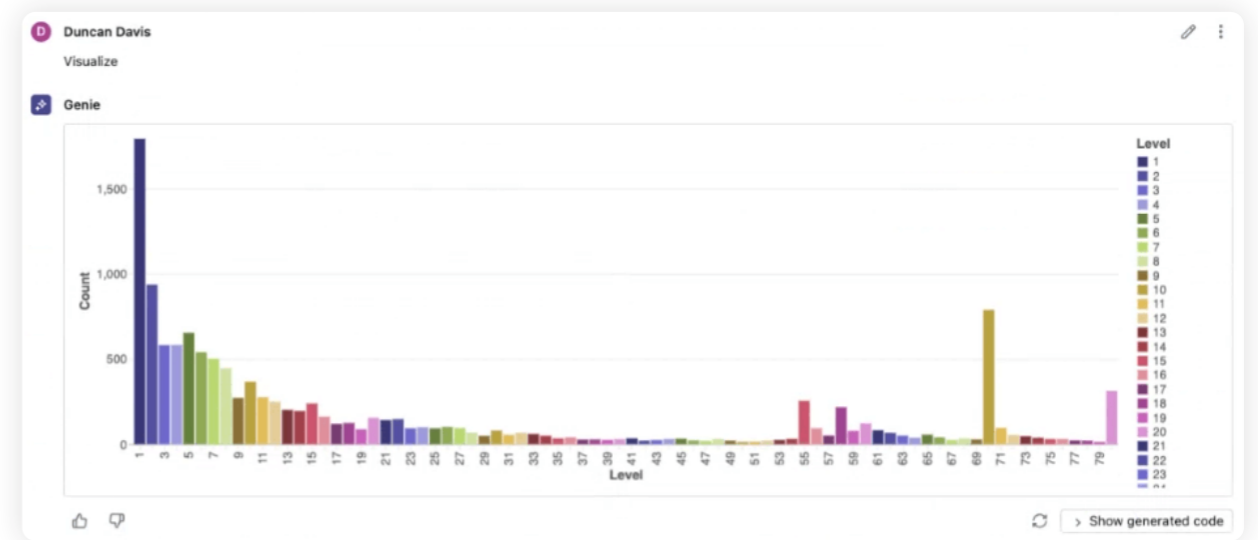
This is super helpful. We immediately see that there's a lot of people creating Blood Elves, but not leveling them up. If we had player data, we might take this as a signal to change the way we're thinking about this. We might ask the space to show the top character class, based on play time, for each player to see if there's preference toward a specific class. As we don't have player data, we'll dig deeper into this visualization instead.

There are bumps at 55, 58, 70 and 80. As a game designer for this title we'll recognize: These were level cap changes due to new releases. This insight could become the basis for a reengagement email campaign. "CHAR_NAME the CHAR_CLASS has more adventures ahead of it. Experience NEW_CONTENT on your way to the new level cap of 80, you're only (LEVEL_CAP – CURR_LEVEL) away. Here is FREE_NEW_CONTENT to help you on your journey. You can earn a HIGH_TIER_CONTENT if you reach LEVEL_CAP by EVENT_TARGET_DATE."

Conclusion

This is just a brief example of how a Genie space could help a game designer explore their data, find insight and then take action on it. We hope it has inspired you to try it on one of your datasets. This example focused on player engagement data but the same process and capabilities could be applied to all game domains, including marketing, campaign optimization, server operations, match outcomes and LiveOperations.

Following are testimonials from an AA console game maker and a mobile studio:

- "A lot of the requests I receive are ad hoc — can you find this number for me? These requests end up iterative in nature because I'll give them the answer and then they want to know the why behind the number. Looking at this tool I could put together a Genie space with data related to their question and give them the power to find the answer and then see the details behind it. Not only will this save my team's time but it should also delight my end users." — Data Analytics Director, AA console game maker

- "I have end users who understand the data we're collecting really well, some of them know a bit of SQL, though they aren't experts, but most of them don't. These users, typically product managers, game designers or LiveOps teams, want more access to data and the ability to find answers on their own. With this capability I could create a space focused on their domain and provide them a self-service tool that will speed their ability to react to change and free my team up for bigger projects." — Director of Data Operations, mobile game

With AI/BI Genie spaces, you empower your teams to have a conversation with their data resulting in emergent insight, higher satisfaction from your internal customers and more time for your data teams to work on high-value projects. The information in this section shows how easy it is to create a Genie space. Create one with your own dataset, and learn something new about your players in the process.

If you'd like to learn more about Genie spaces, how to activate them and how to use them, check out the official documentation or reach out to your Databricks account team. For insight into additional games data and AI use cases, see our other Solution Accelerators and our Guide to Game Data and AI.

databricks

06

Success Stories: Real Results on Databricks

databricks

> Our focus to optimize price/performance was met head-on by Databricks. The lakehouse helped us reduce costs without sacrificing performance across mixed workloads, allowing us to optimize data and AI operations today and into the future.

— MOHIT SAXENA
   Co-founder and Group CTO,
   InMobi

SECTION 6.1

InMobi: Driving Meaningful Connections Between Customers and Brands

When it comes to advertising, consumers want to see content that's relevant and personalized — especially if it's appearing on their mobile devices, where time is a valuable resource — so it's important to instantly capture the consumer's attention and drive engagement. InMobi does this by using real-time customer data to deliver targeted mobile advertising and lock screen experiences. But as data processing requirements increased to 20+ terabytes per hour, the cost of running their multicloud data warehouse skyrocketed, while its proprietary nature created silos that hindered collaboration and data sharing.

InMobi migrated from their multicloud data warehouse to Databricks to unify their various workloads (data warehousing, AI and analytics), streamline operations and free up their engineers to work on more valuable tasks, helping them to achieve better operational agility and efficiency. Since moving to the lakehouse, the company has not only significantly reduced their total cost of ownership (TCO) compared to when they used a multicloud data warehouse — they have also improved productivity across the organization, resulting in faster time-to-market of new products.

32%	**15%**	**20%**
LowerTCO compared to their multicloud data warehouse	Faster queries compared to their multicloud data warehouse	Improved performance of supplier reporting

databricks

Complex legacy infrastructure and multicloud data warehouse unwieldy to manage

InMobi is all about targeted advertising and helping brands reach and engage consumers in a meaningful and cost-effective way. But to accurately deliver relevant ads, you need data — a lot of it. Over time, they extended their on-premises Hadoop system by adding multiple cloud data warehouses to address various problems. However, as the amount of data the company needed to process increased exponentially (20TB of data per hour), InMobi continued to build on top of their legacy system, resulting in a multicloud data warehouse that had a host of challenges: It was overly complex, had outages, was extremely costly as they scaled, and led to data silos that limited data sharing and collaboration. The team at InMobi realized that if they continued with this system in place, it would slow down their ability to innovate and keep precious engineering resources locked into maintenance mode.

"The data infrastructure that we built worked, but it created significant complexity and overhead that pulled our focus away from our own core products," said Madhan Sundaram, Senior Director of Platform Engineering at InMobi. "We needed our talented engineers to create more value for our customers, and to do that, we needed a simpler and more unified system."

What InMobi wanted was a single system that could solve multiple problems. To accomplish this goal, the company needed to consolidate their disjointed systems into a single platform to free up engineers to focus on higher-value tasks such as developing ML and large language models. That's why the team looked to the Databricks Data Intelligence Platform to unify their data warehousing and AI workloads on a single platform.

Migration to lakehouse results in unified data, analytics and AI

Although InMobi has a team of fully capable engineers, they learned a valuable lesson in productivity and operational agility. "We found that we were spending more time maintaining our environment, which was hurting our ability to support and collaborate with the business," explained Sundaram. "We wanted to be more strategic and identify ways to operationalize data more efficiently." After careful evaluation of their current multicloud data warehouse and the prospects of building in-house, they determined that the Databricks Data Intelligence Platform clearly aligned best with their goals — to improve developer productivity through reduced infrastructure complexity while achieving the best possible price/performance.

Once they made their selection, the team partnered with Databricks to start the migration planning process. With over a decade of customizations built on top of existing systems and over 1 petabyte of data, InMobi knew the migration was going to be complex. On the ETL side alone, there were 150 pipelines and eight teams involved with migrating from open source Apache Spark™. They also needed to migrate approximately 300 reporting dashboards to ensure there was no disruption of information delivery to suppliers and customers. To help navigate this process, Databricks worked closely with InMobi — which had allocated two in-house engineers per team to provide support — and Databricks' implementation partner Celebal Technologies on the optimizations needed for a seamless migration.

As a result, Databricks was able to help InMobi migrate from their multicloud data warehouse to the lakehouse with ease. "Databricks gives us the edge in terms of being able to utilize industry-first features that help us establish technical differentiation," said Sundaram. "Their expertise came through during the migration as they helped us optimize our compute resources for cost and performance. The ownership that Databricks took to drive the optimizations and how transparent and educational they were was commendable."

databricks

Bringing more personalized mobile ads to customers everywhere

Now, with a unified, more streamlined lakehouse architecture, InMobi is able to take full advantage of their robust customer data to deliver smarter, more personalized mobile advertising. Various teams leverage Databricks notebooks for ad hoc analysis, Power BI for visualizations on top of Databricks SQL — the serverless data warehouse on the lakehouse — and MLflow to build their next-generation AI platform. They've also found great success with Delta Live Tables for anomaly detection, with a 50% improvement in SLAs and an 80% reduction in costs. Data silos and data discoverability are no longer an issue either, thanks to Unity Catalog. With Unity Catalog, they can now govern access at the table and column levels, while ensuring complete data lineage is captured to ensure they have visibility into where data comes from and whether it's stale or not. With a platform designed to meet their analytics and AI needs, the InMobi team is diving into new technologies, including large language models (LLMs), to help deliver insights to their customers more efficiently. "We've started to look at implementing LLMs to make it easier for our end users to ask a question and find the information they need. The lakehouse architecture will make this effort easier, as jobs will run automatically under the hood. This will give our teams the ability to simply ask a question without any expertise and get the contextual answers they want at their fingertips," explained Sundaram.

In terms of business impact, there've been a number of measurable improvements across the board after the migration. Not only are infrastructure costs 34% lower than before, but there is also a 15% boost in query speeds and 20% fewer job failures compared to their previous data environment, all contributing to 20% better performance of reporting and insights delivery to end users. The TCO is 32% lower compared to when they used a multicloud data warehouse, and there's been a 24% cost reduction in running their ETL pipelines as well. More qualitatively, the team is seeing overall better reliability — with systems more stable than ever — and experiencing a positive reputation boost with their customers.

> **"Our rate of experimentation has improved tremendously," said Mohit Saxena, Co-founder and Group CTO at InMobi. "Databricks has simplified collaboration, and the unified approach that the lakehouse offers allows us to be more efficient, productive and compliant when delivering new features and products."**

By moving to a unified platform on the Databricks Data Intelligence Platform, InMobi can now focus on innovating in the mobile advertising space to deliver real-time personalization that drives value for both InMobi's customers as well as their internal end users.

> Read More Success Stories

databricks

SECTION 6.2

Akamai: Delivering Real–Time Analytics at Scale With Delta Lake

Akamai runs a pervasive, highly distributed content delivery network (CDN). Its CDN uses approximately 345,000 servers in more than 135 countries and over 1,300 networks worldwide to route internet traffic for some of the largest enterprises in media, commerce, finance, retail and many other industries. About 30% of the internet's traffic flows through Akamai servers. Akamai also provides cloud security solutions.

In 2018, the company launched a web security analytics tool that offers Akamai customers a single, unified interface for assessing a wide range of streaming security events and perform analysis of those events. The web analytics tool helps Akamai customers to take informed actions in relation to security events in real time. Akamai is able to stream massive amounts of data and meet the strict SLAs it provides to customers by leveraging Delta Lake and the Databricks Data Intelligence Platform for the web analytics tool.

"

Delta Lake allows us to not only query the data better but to also acquire an increase in the data volume. We've seen an 80% increase in traffic and data in the last year, so being able to scale fast is critical.

— TOMER PATEL
Engineering Manager,
Akamai

‹1
Minute ingestion time,
reduced from 15 min

›85%
Of queries have a response
time of 7 seconds or less

databricks

Ingesting and streaming enormous amounts of data

Akamai's web security analytics tool ingests approximately 10GB of data related to security events per second. Data volume can increase significantly when retail customers conduct a large number of sales — or on big shopping days like Black Friday or Cyber Monday. The web security analytics tool stores several petabytes of data for analysis purposes. Those analyses are performed to protect Akamai's customers and provide them with the ability to explore and query security events on their own.

The web security analytics tool initially relied on an on-premises architecture running Apache Spark™ on Hadoop. Akamai offers strict service level agreements (SLAs) to its customers of 5 to 7 minutes from when an attack occurs until it is displayed in the tool. The company sought to improve ingestion and query speed to meet those SLAs. "Data needs to be as real-time as possible so customers can see what is attacking them," says Tomer Patel, Engineering Manager at Akamai. "Providing queryable data to customers quickly is critical. We wanted to move away from on-prem to improve performance and our SLAs so the latency would be seconds rather than minutes."

After conducting proofs of concept with several companies, Akamai chose to base its streaming analytics architecture on Spark and the Databricks Data Intelligence Platform. "Because of our scale and the demands of our SLA, we determined that Databricks was the right solution for us," says Patel. "When we consider storage optimization, and data caching, if we went with another solution, we couldn't achieve the same level of performance."

Improving speed and reducing costs

Today, the web security analytics tool ingests and transforms data, stores it in cloud storage, and sends the location of the file via Kafka. It then uses a Databricks Job as the ingest application. Delta Lake, the open source storage format at the base of the Databricks Data Intelligence Platform, supports real-time querying on the web security analytics data. Delta Lake also enables Akamai to scale quickly. "Delta Lake allows us to not only query the data better but to also acquire an increase in the data volume," says Patel. "We've seen an 80% increase in traffic and data in the last year, so being able to scale fast is critical."

Akamai also uses Databricks SQL (DBSQL) and Photon, which provide extremely fast query performance. Patel added that Photon provided a significant boost to query performance. Overall, Databricks' streaming architecture combined with DBSQL and Photon enables Akamai to achieve real-time analytics, which translates to real-time business benefits.

Patel says he likes that Delta Lake is open source, as the company has benefitted from a community of users working to improve the product. "The fact that Delta Lake is open source and there's a big community behind it means we don't need to implement everything ourselves," says Patel. "We benefit from fixed bugs that others have encountered and from optimizations that are contributed to the project." Akamai worked closely with Databricks to ensure Delta Lake can meet the scale and performance requirements Akamai defined. These improvements have been contributed back to the project (many of which were made available as part of Delta Lake 2.0), and so any user running Delta Lake now benefits from the technology being tested at such a large scale in a real-world production scenario.
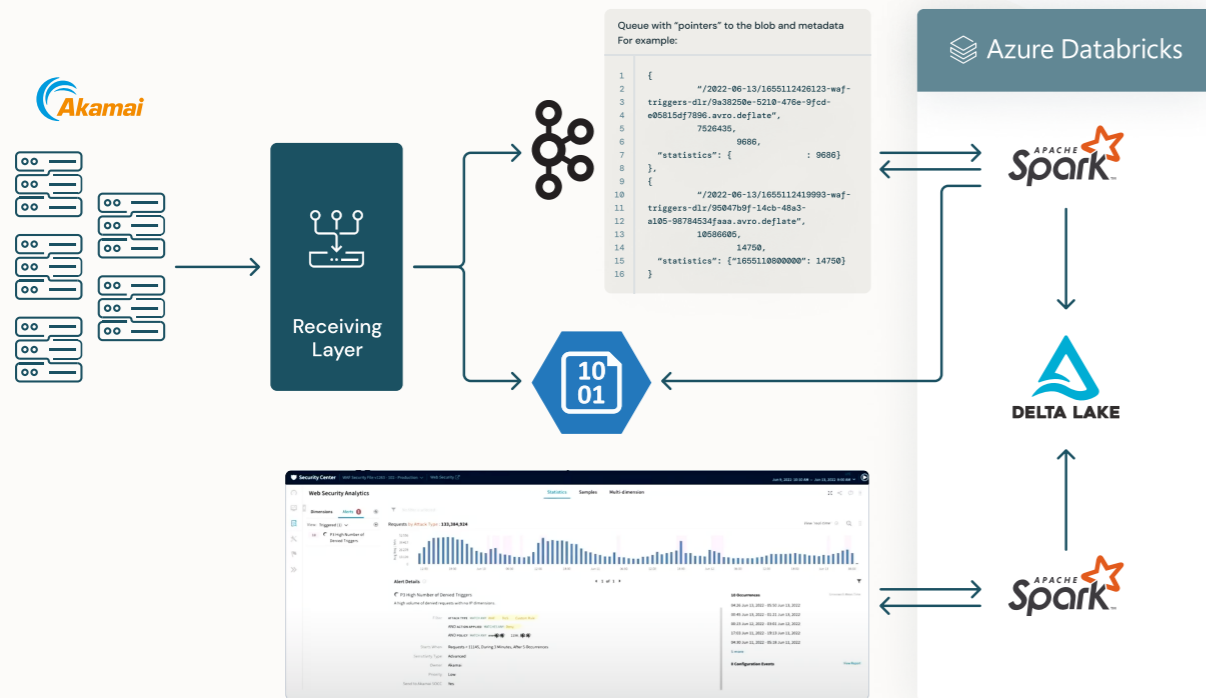
databricks

Meeting aggressive requirements for scale, reliability and performance

Using Spark Structured Streaming on the Databricks Data Intelligence Platform enables the web security analytics tool to stream vast volumes of data and provide low-latency, real-time analytics-as-a-service to Akamai's customers. That way Akamai is able to make available security event data to customers within the SLA of 5 to 7 minutes from when an attack occurs. "Our focus is performance, performance, performance," says Patel. "The platform's performance and scalability are what drives us."

Using the Databricks Data Intelligence Platform, it now takes under 1 minute to ingest the security event data. "Reducing ingestion time from 15 minutes to under 1 minute is a huge improvement," says Patel. "It benefits our customers because they can see the security event data faster and they have a view of what exactly is happening as well as the capability to filter all of it."

Akamai's biggest priority is to provide customers with a good experience and fast response times. To date, Akamai has moved about 70% of security event data from its on-prem architecture to Databricks, and the SLA for customer query and response time has improved significantly as a result. "Now, with the move to Databricks, our customers experience much better response time, with over 85% of queries completing under 7 seconds." Providing that kind of real-time data means Akamai can help its customers stay vigilant and maintain an optimal security configuration.

 Read More Success Stories

SECTION 6.3

Quartile: Becoming the Largest e-Commerce Ad Platform

Quartile is the world's largest e-commerce cross-channel advertising platform, serving over 5,000 connected ad accounts from more than 10 channels. Built on six patented machine learning technologies, the platform automates and optimizes e-commerce advertising on Google, Facebook, Amazon, Instacart, Walmart and more. Quartile pairs leading technology with marketing experts who create strategies that are tailored to their customers' business goals. Thousands of sellers worldwide trust Quartile's funnel optimization approach to unlock the full potential of their selling and advertising across multiple channels, entirely integrated and at scale.

"

The Databricks Data Intelligence Platform has been enabling our technology teams to keep improving our time to market on new solutions and delighting customers with quality data sets. Becoming the largest e-commerce cross-channel ad platform would be much harder without Databricks and the lakehouse architecture.

— DANIEL KNIJNIK
    CEO,
    Quartile

80%
Reduction in data storage from traditional databases

10x faster
45 minutes to optimize bidding, down from 7.5 hours

databricks

Size of data and performance needed to scale

Quartile faced challenges in storing and processing data for multiple advertising channels due to crucial needs in reporting and consolidating sales data, including over 60 days of attribution. The previous architecture couldn't keep up with the size of data Quartile was processing. The team was batch processing over 10TB of data on a single job, which applied all transformations required for data reporting, causing server unavailability and late deliveries of data points. This process alone, which is responsible for improving the performance of ads, had severe performance problems, such as individual jobs running up to 7.5 hours every day.

Technology evolution, from legacy to modern data stack

As part of the evolution of the company, Quartile's data architecture has reached new levels of maturity, growing from traditional SQL databases running on Azure cloud to a new solution with the Databricks Data Intelligence Platform as the foundation. This modernization has had direct impact in several areas of the company and clear benefits for Quartile's customers: with Delta Lake there has been more data reliability, faster performance and reduced costs. When migrating the data from a traditional SQL database to Databricks, they saw a considerable reduction in data volume, mainly due to the optimizations of Delta with version control and Parquet compacting, resulting in storage reduction from 90TB to about 18TB.

The lakehouse architecture has enabled Quartile's data stack to evolve. This has been accomplished through several processes: by leveraging Databricks Auto Loader for incremental and efficient processing of new data files as they arrive in cloud storage; by using Delta Lake for its open format storage layer, which delivers reliability, security and performance on the data lake; by using Databricks SQL, which continues to bring data engineers and data analysts closer with easy-to-build queries and dashboards; and with Databricks Workflows, which connects all of the pieces together in a stable and scalable way.

To provide the best experience for customers, Quartile needs to be able to retrieve accurate data. This is an important challenge that is easier to handle with Spark User-Defined Functions, as the company uses the power of parallelism to break down processing into as many parts as needed. For their solution to scale, they utilize Terraform for deploying all workspaces, easily spinning up new clusters and jobs as well as ensuring the correct standards are being followed across the company.

Helping their partners run ads better

It is critical that Quartile's customers have a centralized solution in which they can analyze their sales, costs and other metrics related to their campaigns. At Quartile, they take advantage of the solid data engineering work and integrate Databricks with Power BI for embedding the dashboards directly in their portal. This helps provide a single place for clients to both configure marketing campaigns across several channels as well as follow up on performance changes while still maintaining a smaller data storage footprint that is 80% less expensive compared to the data stored in traditional data warehouses by leveraging Delta Lake's file format on object storage. The ability to combine data from all different channels has already helped several of their customers — SmartyPants, as an example, has grown over 100% since they partnered with Quartile.

databricks

But that's not all. Quartile has also patented algorithms for improving the performance of ads, which are implemented utilizing the machine learning persona in Databricks. The ability of having a central lakehouse architecture to build their entire data stack has made the lives of Quartile's developers much simpler, allowing them to focus on developing innovative solutions and bringing increasingly better results for their customers. As another example, OfficeSupply has had excellent results during their first year of working with Quartile, with a 67% increase in Google Ads revenue and a 103% increase in Google Shopping clicks for trademark terms by improving the performance of individual jobs — they used to take 7.5 hours but now run in 45 minutes on the lakehouse architecture.

Looking ahead, Quartile is continuing to partner with Databricks to build their modern data stack, integrating and testing newer solutions. This includes Delta Live Tables for higher data quality checks, Delta Sharing to send customers their own data, and Data Marketplace to enable clients to get started quickly. Quartile has a bold target to develop the first cross-channel learning for an ads optimization algorithm in this space, and Databricks will be the center of these innovations.

databricks

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.

START YOUR FREE TRIAL

Contact us for a personalized demo
databricks.com/company/contact

databricks