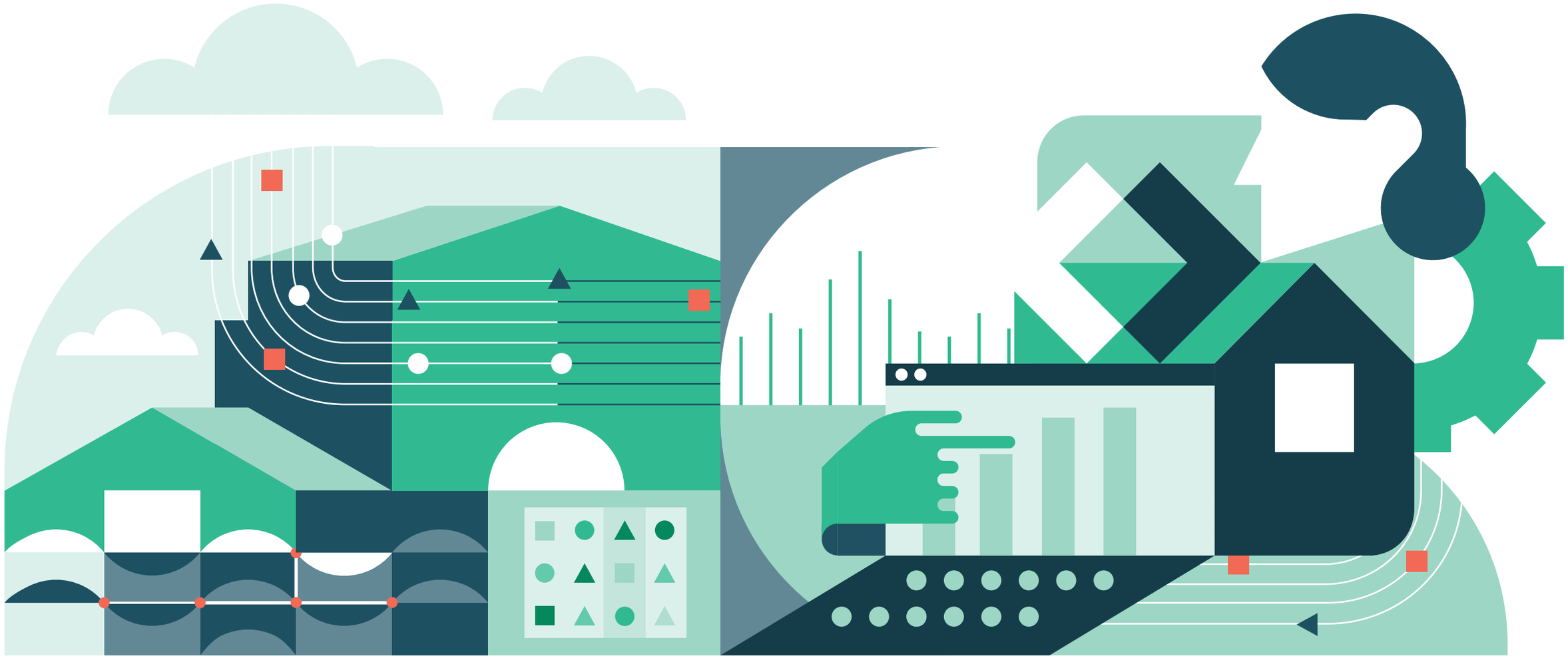
A Guide to Data Warehousing in the Lakehouse

Databricks SQL for data warehouse practitioners



databricks

Contents

Introduction 4

CHAPTER 1 **What Is Databricks SQL?** 5

The Databricks SQL architecture 7

The well-architected lakehouse 9

The rise of the medallion mesh 12

Data modeling 15

Why Databricks SQL Serverless is the best for BI workloads 18

CHAPTER 2 **Use Cases** 20

Building data applications 21

Sensitive data governance design patterns in Databricks SQL 23

Dynamic SQL design patterns with Databricks SQL on workflows 26

End-to-end streaming ELT on Databricks SQL with streaming tables and materialized views 29

Enabling data analysts with dashboards on your data warehouse 32

CHAPTER 3 **End-to-End Steps** **35**
Ingest and transform 36
Enable AI-powered SQL querying 41
Easily build dashboards with Databricks AI/BI 44
Integrating with external platforms 45
Data applications 47
Data sharing and monetization 47

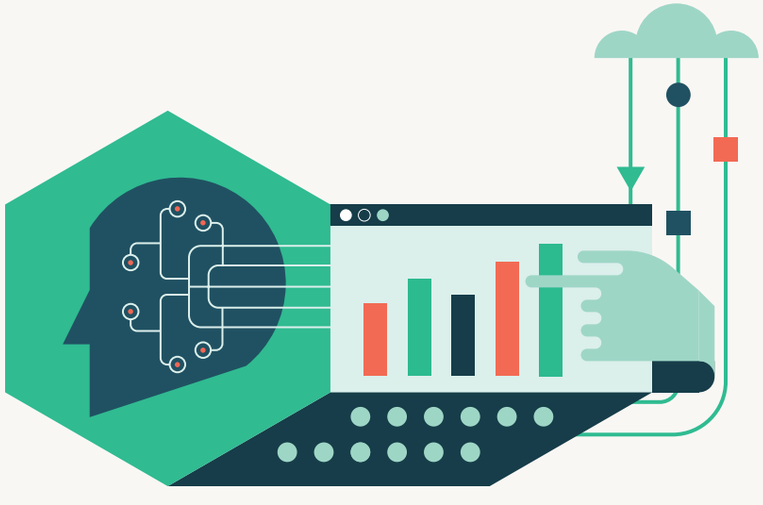
CHAPTER 4 **Migrating to Databricks SQL** **48**
Migration process 50
Migration strategy 52
Migration tools 53
Next steps 54

Introduction

Despite decades of technological advancements, data warehouse management continues to face three primary challenges. First, governance is complex due to the mix of tools, AI models and dashboards organizations employ, making it difficult to control and manage data access across various platforms. Second, performance optimization is labor-intensive, often requiring constant manual adjustments. This leads to frustrated users and delayed reports. Finally, ease of use is a persistent issue. Many data warehouse tools are outdated and require specialized knowledge, limiting who can use them. This lack of user-friendly interfaces prevents many in an organization from easily accessing and analyzing data, creating a barrier to widespread data utilization. Databricks SQL was developed to solve these challenges.

Databricks SQL is the intelligent data warehouse built on the Databricks Data Intelligence Platform. It addresses the complex data needs of modern organizations by democratizing analytics for both technical and business users alike, and it leverages a data intelligence engine that understands the uniqueness of your data to provide intelligent experiences and predictive optimizations.

In this guide, we'll dig deeper into how Databricks SQL is revolutionizing data analytics and enabling organizations to transform data into actionable insights quickly and efficiently. And, we'll explore real-world examples of how businesses are leveraging Databricks SQL to drive innovation.



CHAPTER

01

What Is Databricks SQL?

Databricks SQL is the intelligent data warehouse built on the Databricks Data Intelligence Platform. It represents a paradigm shift in data warehousing to data lakehouse architecture by combining the best elements of traditional data warehouses with the flexibility and scalability of modern cloud architecture, while adding the power of artificial intelligence. It enhances the capabilities of the Databricks Data Intelligence Platform by facilitating data transformation and analysis for a wide range of users, from business intelligence (BI) analysts and data architects to data engineers.

Databricks SQL also uses Databricks Data Intelligence to automatically optimize the platform to provide an excellent price-to-performance ratio.

Databricks SQL excels in five key areas:

- **Fast time to insights:** Databricks SQL allows users to access data using plain English to automatically create SQL queries. This feature streamlines query refinement and is available to everyone in the organization, significantly reducing the time from data to decision.
- **Price/Performance:** Through serverless compute combined with AI-optimized processing, Databricks SQL achieves top-tier performance and scale at lower costs. AI-driven performance optimizations like Predictive I/O deliver leading performance and cost savings without requiring manual tuning.
- **Unified governance:** Databricks SQL incorporates advanced provisions for data governance, security and compliance. By incorporating Unity Catalog, Databricks SQL establishes a unified governance layer across all data and AI assets, regardless of their location, empowering organizations to uphold data quality, implement across restrictions, oversee data activities, safeguard sensitive data and adhere to regulatory standards.
- **Reduced complexity:** Databricks SQL unifies all data, analytics and AI on a single platform that supports SQL and Python, notebooks and IDEs, batch and streaming data and all major cloud providers. The Lakehouse Federation feature enables data and AI teams to use Databricks SQL to run queries across multiple external data sources, right within Databricks.
- **Rich ecosystem:** The platform provides a collaborative workspace where data professionals can quickly exchange knowledge. Users can leverage SQL and tools such as PowerBI, Tableau, dbt, Fivetran, Qlik, Informatica and Alteryx with Databricks for BI, data ingestion and transformation.

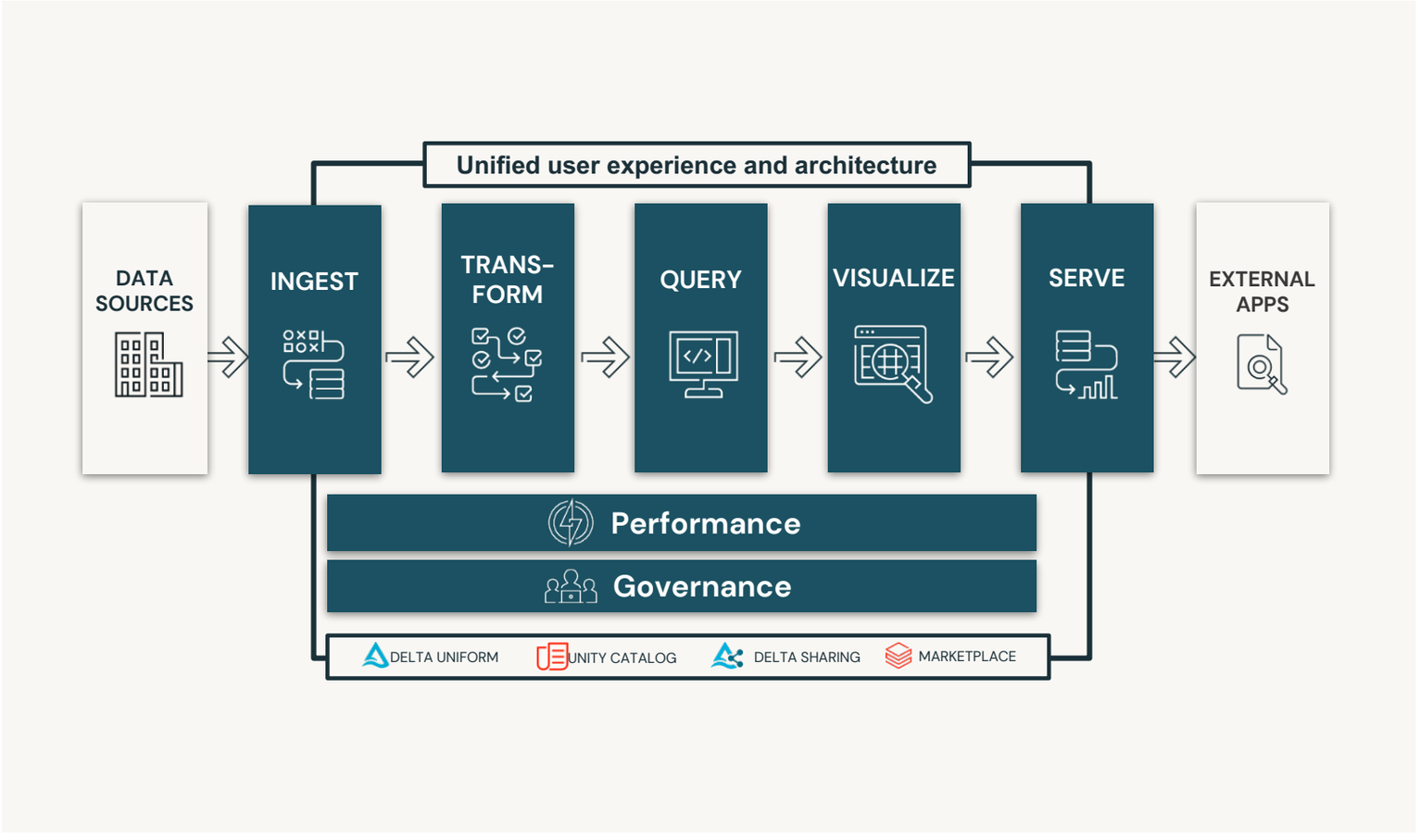
Finally, Databricks SQL has tapped the power of AI to address the data warehouse's biggest historical challenges — performance, governance and usability — thanks to a deeper understanding of your specific data and how it's used. Ultimately, Databricks SQL is more than just a data warehouse — it's an intelligent, efficient and accessible platform that enables organizations to innovate faster, make data-driven decisions more quickly and drive better business outcomes.

The next section takes a deeper dive into the Databricks SQL architecture.

The Databricks SQL architecture

Databricks SQL uses the data lakehouse architecture. We'll go into more detail on a well-architected data lakehouse in a bit. The data lakehouse architecture reinforces the commitment to open table formats, so you don't have to worry about locking in on a format.

Within this unified user experience and architecture, we have capabilities for ingest, transform, query, visualize and serve. A wide variety of data sources can be used with Databricks SQL, including unstructured and streaming data. The sources can be in different clouds, different platforms and different formats.



What is the scope of the Databricks Data Intelligence Platform?

The Databricks Data Intelligence Platform is a unified data and AI cloud service that covers a wide range of data-related activities:

- ELT (extract, load, transform) processes
- ETL (extract, transform, load) processes
- Machine learning and AI
- Data warehousing
- Business intelligence

The scope of the lakehouse extends beyond traditional data warehouses or data lakes, aiming to provide a single, unified platform for all data-related workloads. It encompasses data storage, processing, analysis and governance, offering a comprehensive solution for modern data architectures.

What is the vision of the well-architected lakehouse?

The vision of the well-architected lakehouse can be summarized in six key principles:

- Curate data and offer trusted data as a product (DaaP)
- Remove data silos and minimize data movement
- Democratize value creation through self-service experience
- Adopt an organization-wide data governance strategy
- Encourage the use of open interfaces and open formats
- Build to scale and optimize for performance and cost

How does the lakehouse integrate into the cloud provider's and the customer's cloud architecture?

The Databricks lakehouse architecture integrates seamlessly with major cloud providers such as AWS, Azure and GCP.

Integration aspects include:

- Utilizing native cloud storage services (e.g., S3, Azure Blob Storage, Google Cloud Storage)
- Leveraging cloud-native security and identity management services
- Integrating with cloud-specific networking and data transfer services
- Utilizing cloud provider's monitoring and observability tools

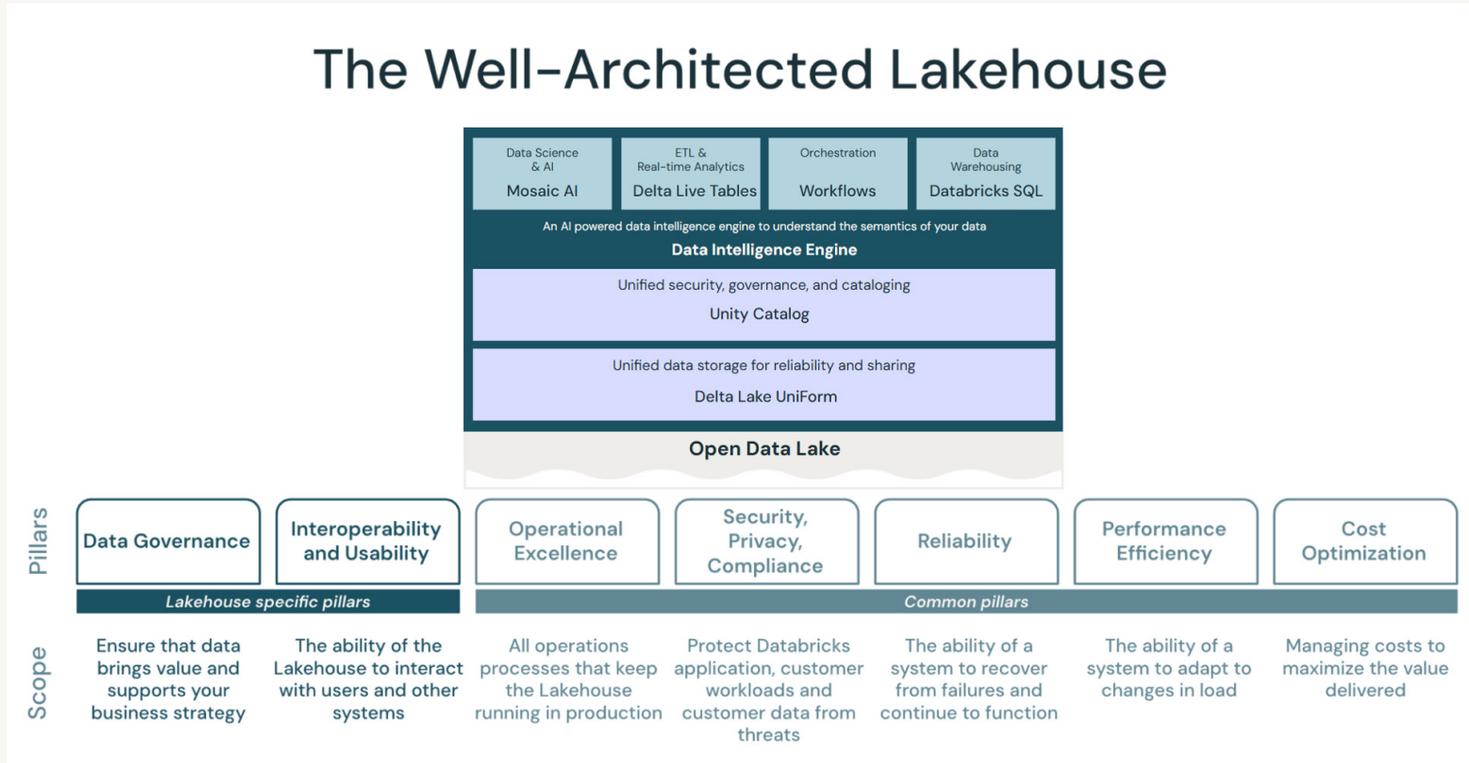
The well-architected lakehouse

What is "good" for the underlying lakehouse architecture?
Databricks' well-architected framework for the lakehouse builds upon the existing well-architected frameworks provided by cloud providers like AWS, Azure and GCP.

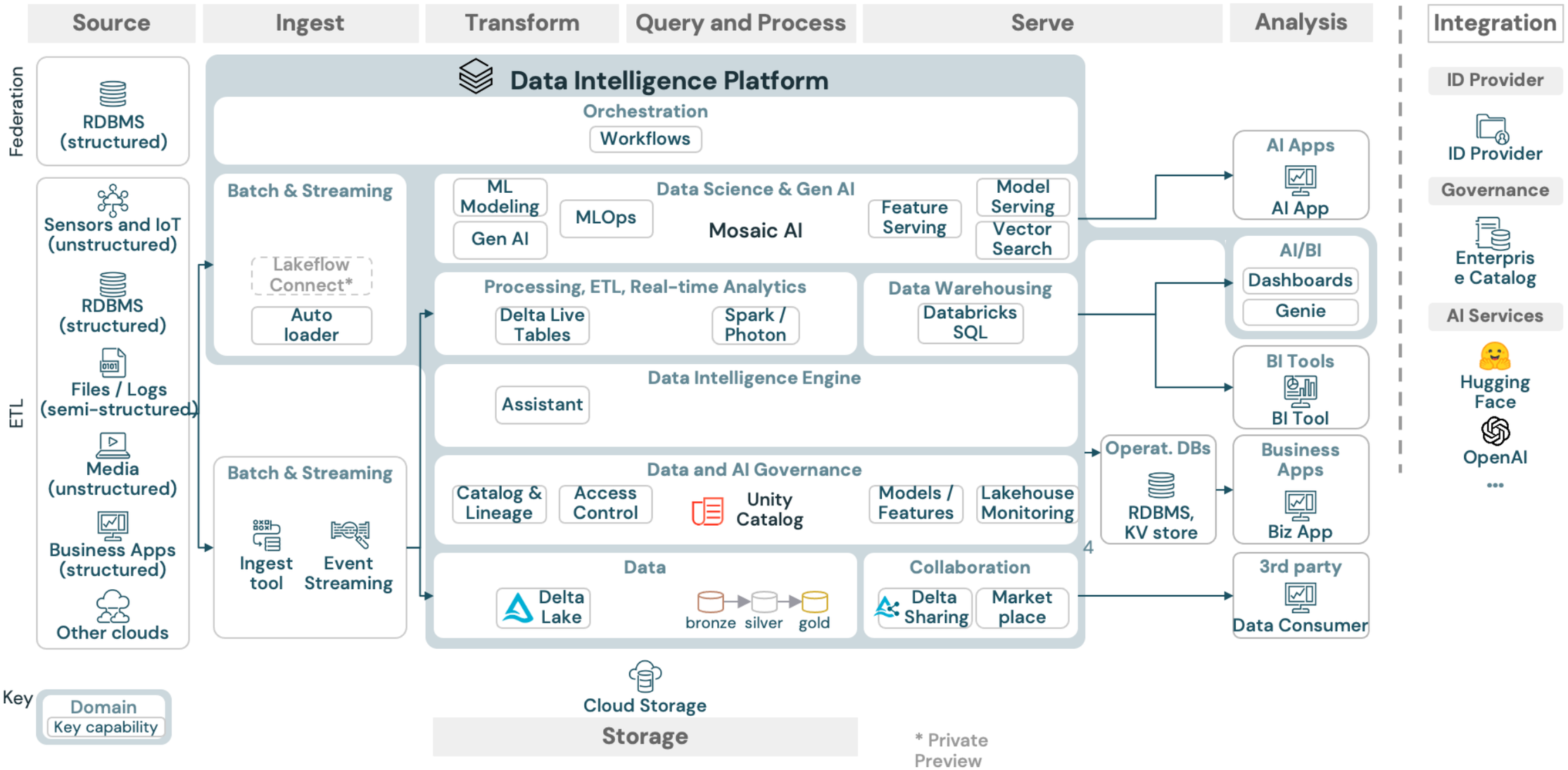
The well-architected lakehouse framework defines seven pillars that describe what is "good" for the underlying architecture:

- Data governance
- Interoperability and usability
- Operational excellence
- Security, privacy and compliance
- Reliability
- Performance efficiency
- Cost optimization

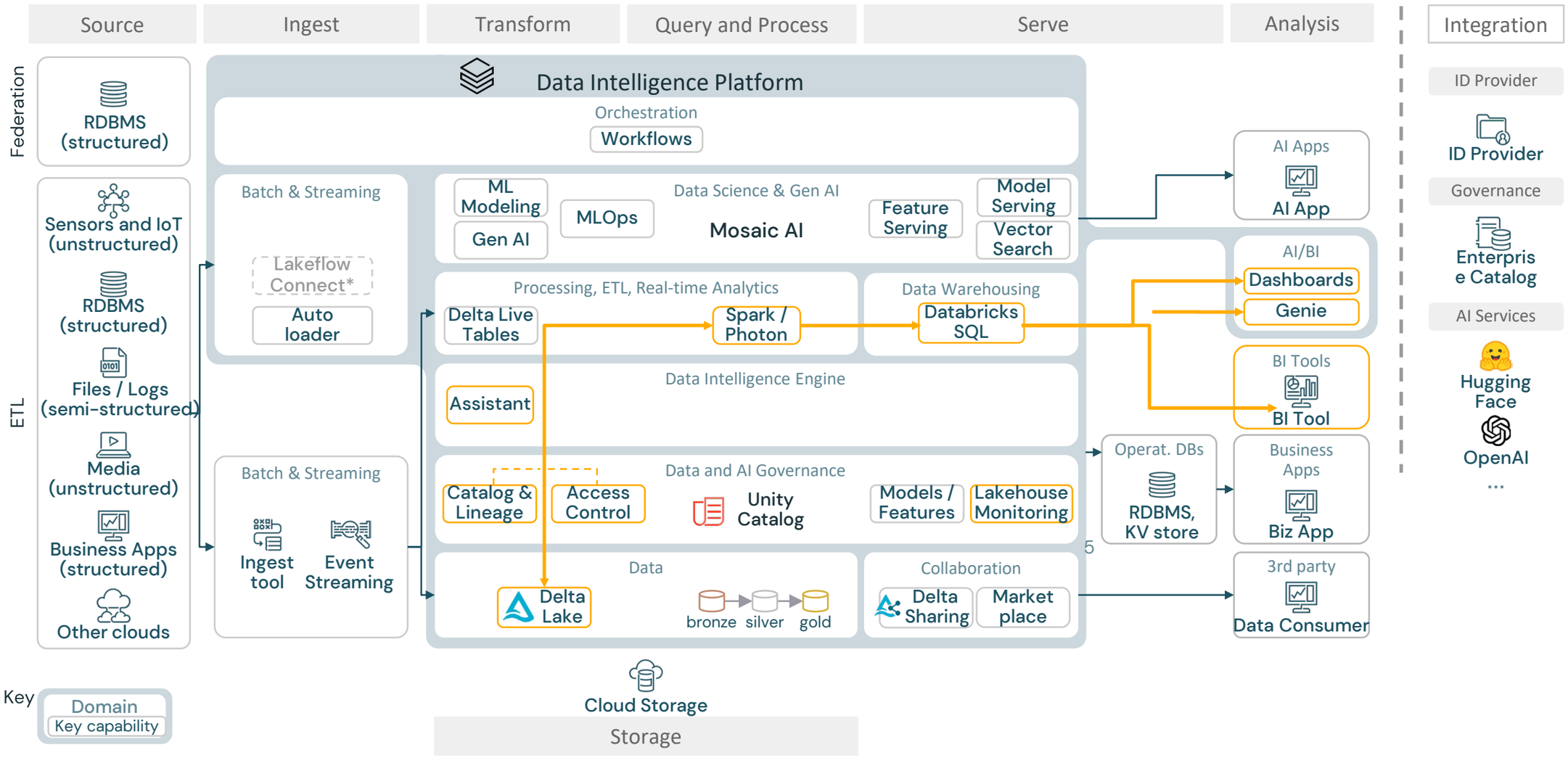
These pillars provide a framework for evaluating and improving the lakehouse architecture. They guide organizations in designing systems that are not only technically sound but also aligned with business objectives and regulatory requirements.



The reference architecture for a well-architected lakehouse below covers architectural components in terms of data source, ingestion, transformation, querying and processing, serving, analysis/output and storage.



In the context of a data warehouse, a simplified DW/BI architecture is shown below, which covers critical components for data warehouse and BI in Databricks Data Intelligence Platform.



The rise of the medallion mesh

The medallion mesh represents an evolution in data management, combining the strengths of several established patterns and practices to address the challenges faced by modern data-driven organizations. This innovative approach emerges from the fusion of Data Mesh principles, medallion architecture and tools like dbt, all underpinned by the capabilities of Databricks Delta Lake and Unity Catalog.

The medallion mesh addressed a critical issue highlighted by a SnapLogic study: IT departments and data and AI teams lose an average of four working hours per employee each week due to data preparation issues. This inefficiency grows as organizations grow, requiring a more robust and scalable approach to data management.

The medallion mesh draws from three key components:

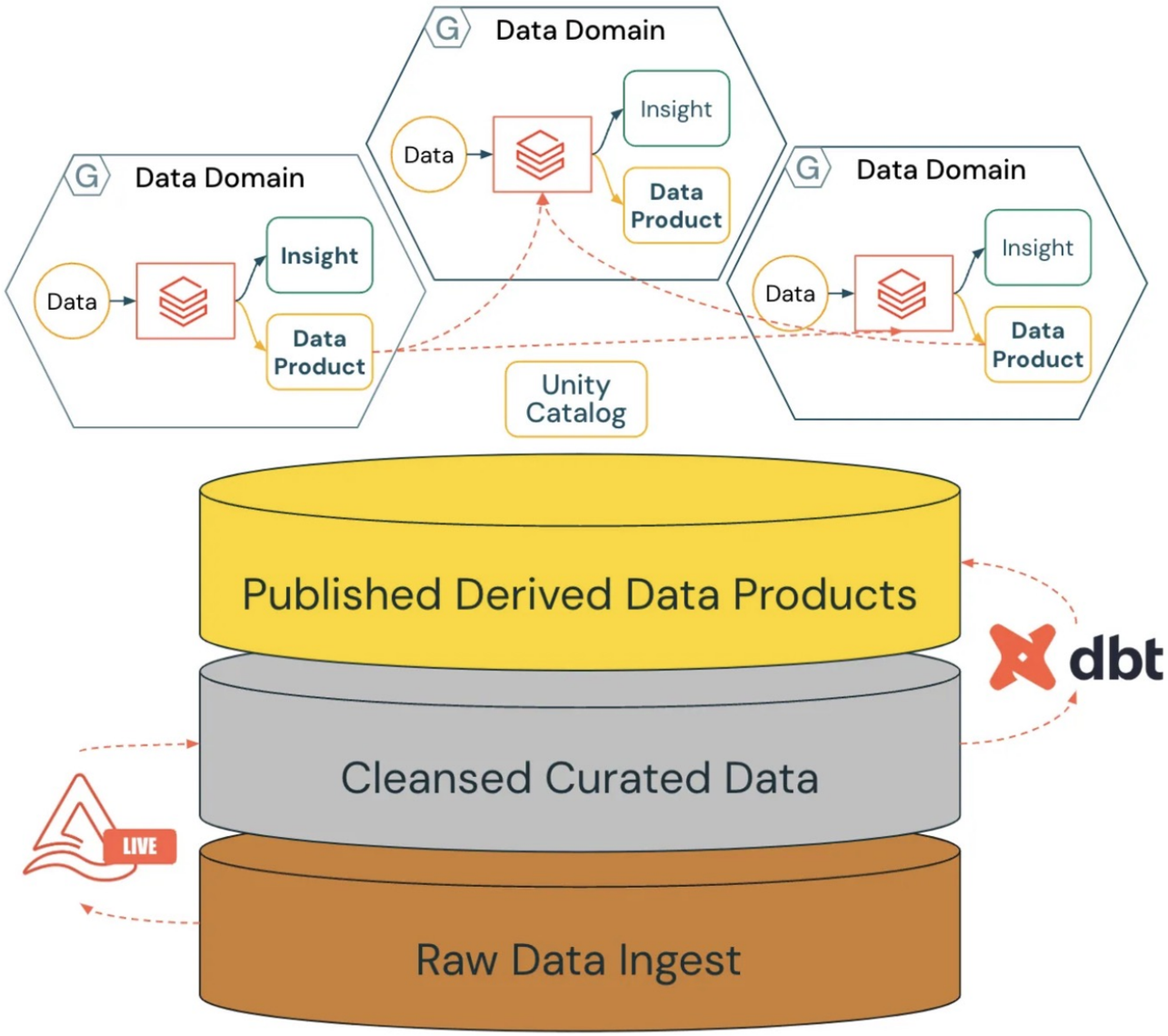
- **Data Mesh:** A decentralized data management framework created by Zhamak Dehghani, fostering a self-service culture around creating data products in a decentralized manner grouped by domain.
- **Medallion architecture:** Emerging from the shortcomings of schema-on-read attempts with Hadoop, this pattern organizes data into Bronze (raw data), Silver (cleansed and structured data) and Gold (derived data products) layers.
- **Data build tool (dbt):** Focusing on the "T" in ELT, dbt handles transformations in data pipelines, enabling version-controlled dynamic SQL in modern cloud data warehouses.

The medallion mesh pattern allows for a decentralized approach to producing data products flexibly in the cloud. Data is stored at rest in abundant and cheap data lakes, with federated just-in-time serverless compute providing a self-service consumption layer for analysts, scientists and engineers.

The implementation of the medallion mesh is further enhanced by specific tools:

- Delta Live Tables (DLT) for Bronze to Silver transformations, preferred by core IT teams and data engineers
- Silver to Gold transformations (e.g., dbt), used by analytics engineers to produce derivative data products
- Unity Catalog for unified governance, providing column-level lineage tracking and enabling decentralized self-service governance

This combination of tools and practices helps organizations to adapt to the ever-challenging world of delivering secure and well-governed data and AI solutions. By empowering data producers to craft and publish their data products with DLT and SQLMesh, while maintaining federated governance through Unity Catalog, organizations can reduce the bottlenecks of traditional centralized systems and create a thriving data ecosystem.



Implementing the medallion architecture with Databricks SQL

Here's the process of building an end-to-end medallion architecture pipeline using streaming tables and materialized views.

Bronze layer:
Incremental data ingestion

```
CREATE STREAMING LIVE TABLE BZ_raw_txs
  COMMENT "New raw loan data incrementally ingested from cloud object storage landing zone"
AS SELECT * FROM cloud_files('/demo/frank_uc/landing', 'json')
```

Dimension table:
Materialized view for reference data

```
CREATE MATERIALIZED VIEW ref_accounting_treatment
  COMMENT "Lookup mapping for accounting codes"
AS SELECT * FROM delta.`/demo/frank_uc/ref_accounting_treatment/`
```

Silver layer:
Data enrichment and quality checks

```
CREATE STREAMING LIVE TABLE SV_cleaned_new_txs (
  CONSTRAINT `Payments should be this year`  EXPECT (next_payment_date > date('2020-12-31')),
  CONSTRAINT `Balance should be positive`    EXPECT (balance > 0 AND arrears_balance > 0) ON VIOLATION DROP ROW,
  CONSTRAINT `Cost center must be specified` EXPECT (cost_center_code IS NOT NULL) ON VIOLATION FAIL UPDATE

)
  COMMENT "Livestream of new transactions, cleaned and compliant"
AS SELECT txs.*, rat.id as accounting_treatment FROM stream(LIVE.BZ_raw_txs) txs
  INNER JOIN live.ref_accounting_treatment rat ON txs.accounting_treatment_id = rat.id
```

Gold layer:
Aggregated materialized view

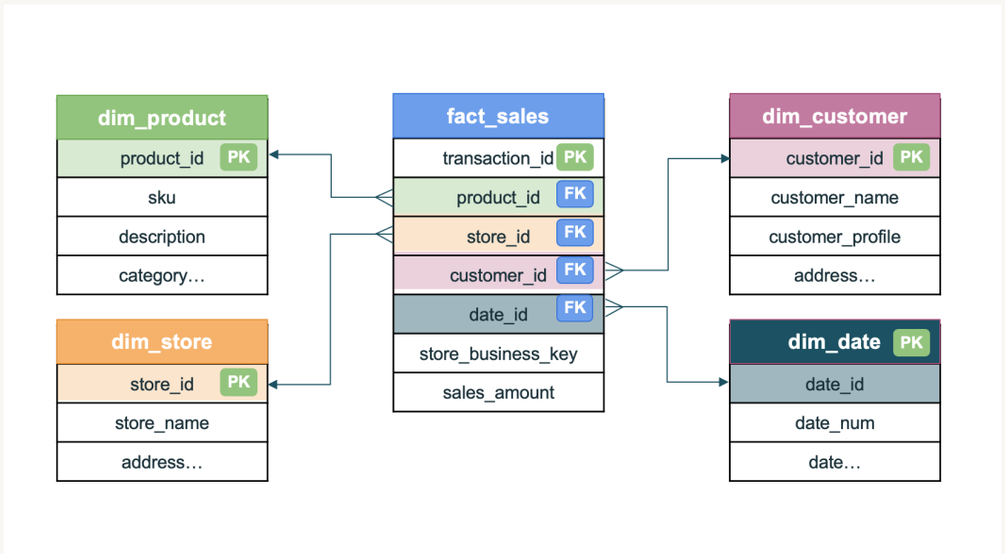
```
CREATE MATERIALIZED VIEW GL_total_loan_balances_1
  COMMENT "Combines historical and new loan data for unified rollup of loan balances"
  TBLPROPERTIES ("pipelines.autoOptimize.zOrderCols" = "location_code")
AS SELECT sum(revol_bal)  AS bal, addr_state   AS location_code FROM live.SV_historical_txs  GROUP BY addr_state
  UNION SELECT sum(balance) AS bal, country_code AS location_code FROM live.SV_cleaned_new_txs GROUP BY country_code
```

Data modeling

Data modeling is a fundamental process in designing and organizing data structures to support efficient storage, retrieval and analysis of information. In cloud-based ecosystems like Databricks, effective data modeling is crucial for organizations aiming to unlock the full potential of their data assets. This section explores key concepts, best practices and tools for data modeling in modern data platforms, with a focus on the Databricks environment.

The star schema: A foundation for data warehousing

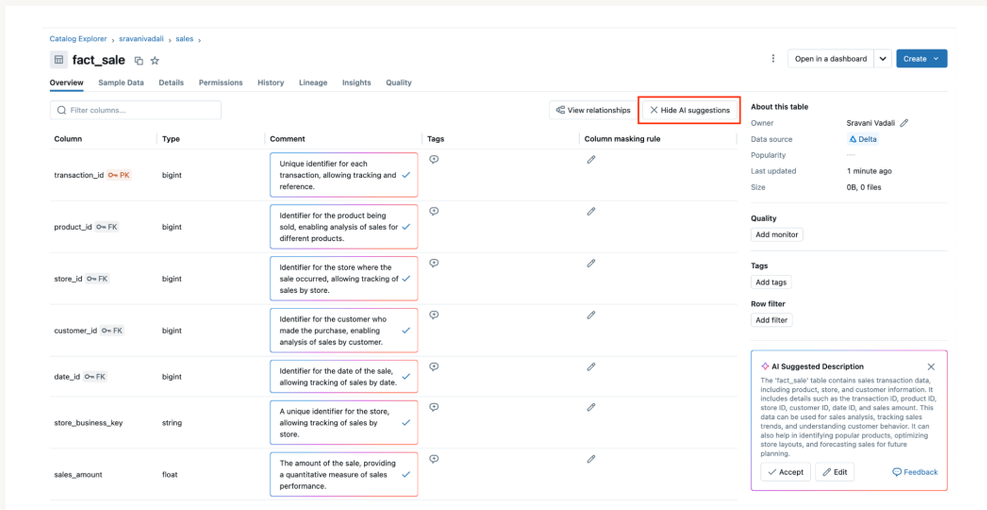
One widely used approach in data warehousing is the star schema. This design pattern consists of a central fact table surrounded by dimension tables, allowing for efficient querying and analysis of transactional data. Key characteristics of the star schema include fact tables and dimension tables.



Implementing star schema in Databricks SQL

When implementing a star schema in Databricks SQL, consider the following best practices:

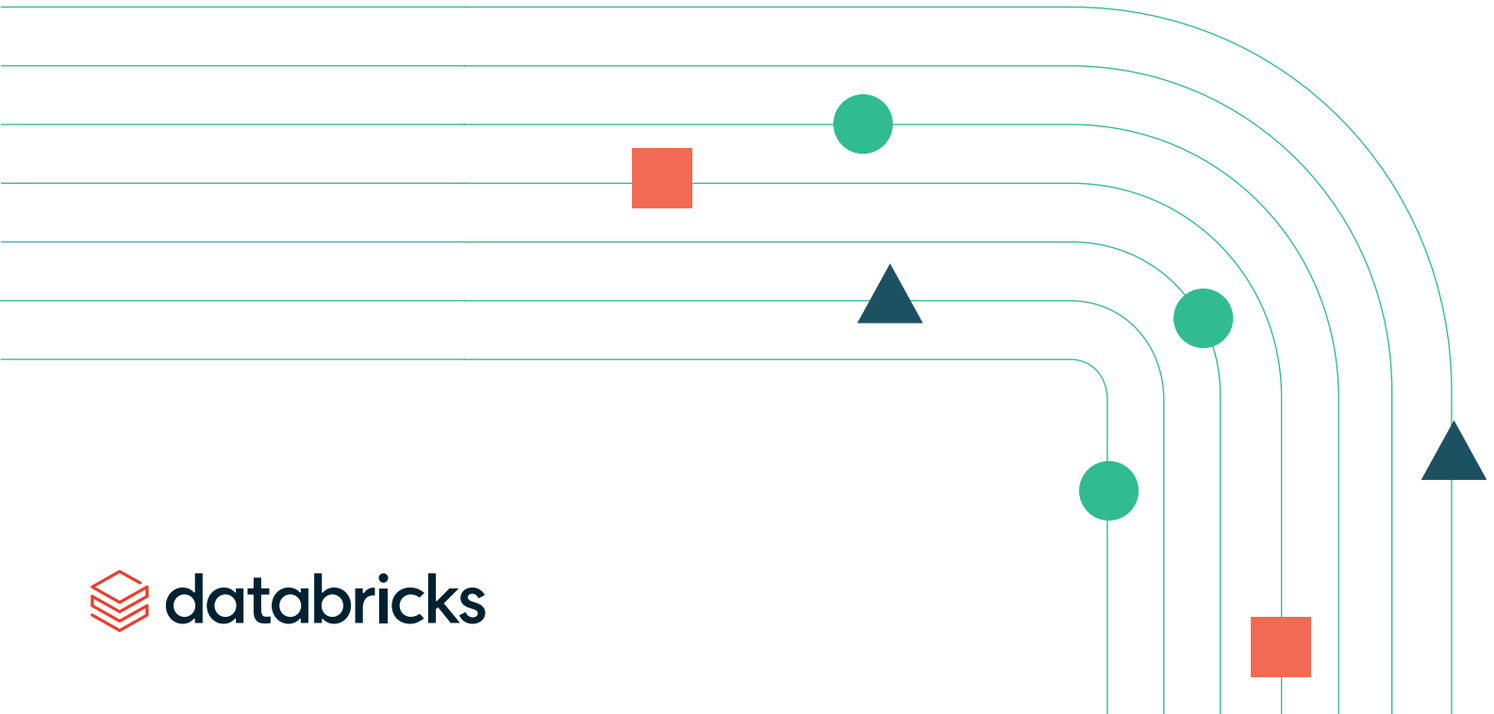
- Use managed Delta Lake tables for both fact and dimension tables
- Implement surrogate keys using *Generated as Identity* columns or hash values
- Utilize Liquid Clustering based on frequently filtered attributes for improved query performance
- Define appropriate constraints (e.g., *Primary Key, Foreign Key*) for data integrity and query optimization
- Leverage Delta Lake features like Time Travel for historical data access
- Document tables and columns using comments and tags for enhanced data governance



Best practices for data modeling in Databricks

To maximize the effectiveness of your data modeling efforts in Databricks, consider implementing the following best practices that leverage the platform's unique features and optimize for performance, scalability and maintainability.

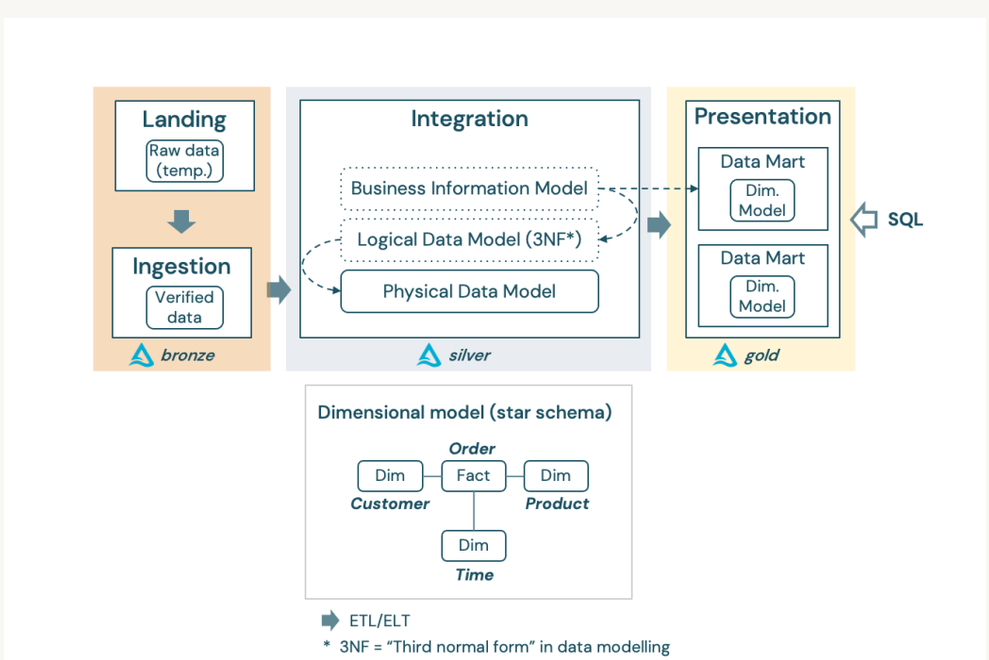
- **Leverage Delta Lake:** Use Delta Lake tables for enhanced ACID compliance and time travel capabilities.
- **Optimize for performance:** Utilize Liquid Clustering and partitioning strategies appropriate for your data and query patterns.
- **Implement data quality checks:** Use streaming table constraints or CHECK constraints to ensure data integrity.
- **Design for scalability:** Consider future growth and query patterns when designing your schema.
- **Document thoroughly:** Use AI-generated comments, tags and external documentation to enhance understanding and maintainability of your data model.
- **Collaborate effectively:** Utilize tools like SqlDBM to facilitate team collaboration and version control.
- **Consider cost optimization:** Design your data model with storage efficiency and query performance in mind to optimize costs in consumption-based environments.



databricks

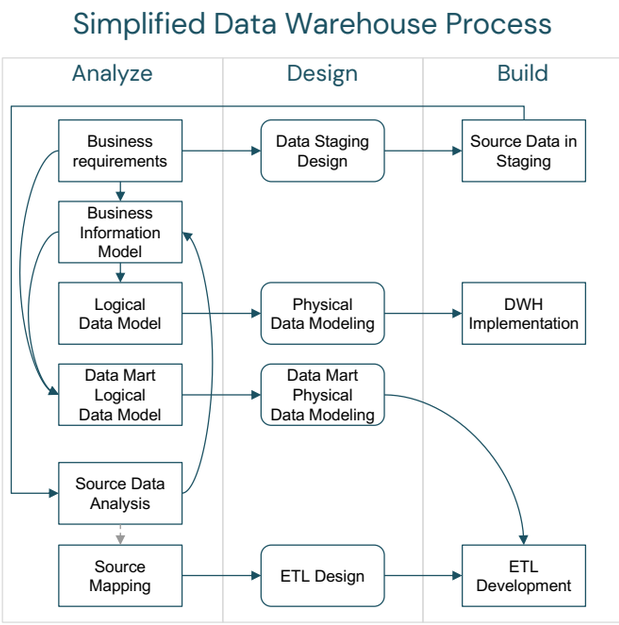
Delta Lake plays a pivotal role in optimizing the performance of these data models. By implementing features such as Liquid Clustering and data skipping, Delta Lake enables faster query performance by organizing data based on frequently accessed columns. Additionally, Delta Lake supports the creation of materialized views and streaming tables, which allow for real-time data processing and the generation of precomputed results that can significantly speed up BI workloads. These capabilities are essential in a lakehouse environment, where the vast amount of data with different varieties requires efficient and flexible data processing and retrieval mechanisms.

Example of dimension modeling



Introducing Materialized Views and Streaming Tables for Databricks SQL

References for data modeling guidance



- Star Schema Data Modeling Best Practices on Databricks SQL
- Data Modeling on Databricks with SqlDBM

Why Databricks SQL Serverless is the best for BI workloads

The ability to quickly retrieve and analyze data is crucial for meeting customer-defined service level agreements (SLAs), objectives (SLOs) and indicators (SLIs). However, companies don't want to spend time managing their infrastructure. For quick startup and elastic scaling — all handled automatically — you can use Databricks SQL Serverless. SQL Serverless offers a combination of performance, scalability and cost-effectiveness that sets it apart from traditional data warehousing solutions. This section explores the key features and benefits that make Databricks SQL Serverless the optimal choice for modern BI workloads.

Disk cache: Enhancing query performance

Databricks SQL Serverless offers advanced caching capabilities, including disk cache. This feature significantly enhances query performance by leveraging locally attached solid state drives (SSDs) to store frequently accessed data.

Cold cache vs. warm cache

Cold cache: When the disk cache is empty, queries must retrieve all data from remote cloud storage, resulting in longer execution times. In our example, a sample query took over 10 seconds to execute with a cold cache.

Warm cache: After preloading data into the cache using the CACHE statement, the same query executed in just two seconds — an 80% improvement in execution time.

Benefits of warm cache

Reduced latency: With 100% of required data available in the disk cache, query execution is significantly faster.

Improved user experience: Faster query responses lead to more responsive BI applications.

Efficient resource utilization: By minimizing the need to fetch data from remote storage, the system operates more efficiently.

Flexibility and scalability

Databricks SQL Serverless also offers flexibility and scalability advantages such as:

Decoupling of storage and compute

Unlike traditional data warehouses, Databricks SQL doesn't store data permanently on the warehouse itself. Instead, data is stored in cloud object stores.

This architecture allows users to size their SQL warehouse independently of data storage, providing greater flexibility in resource allocation.

Serverless architecture

Users can scale computational resources up or down based on query complexity or concurrent query volume without directly impacting data storage costs.

This elasticity ensures optimal performance during peak times and cost-effectiveness during periods of lower demand.

Versatile caching

Databricks cache can be applied to entire tables or specific query results, allowing for fine-tuned performance optimization.

Cost-effectiveness

The serverless nature of Databricks SQL, combined with its efficient caching mechanisms, contributes to its cost-effectiveness and provides the following benefits:

- **Pay-for-use model**

  Organizations only pay for the computational resources they actually use, rather than maintaining a constantly running data warehouse.

- **Optimized resource utilization**

  The disk cache reduces the need for frequent data retrieval from cloud storage, potentially lowering data transfer costs.

- **Scalability without overprovisioning**

  The ability to scale resources on demand means organizations can avoid overprovisioning their data warehouses, leading to cost savings.

Performance at scale

Databricks SQL Serverless is designed to handle large-scale BI workloads efficiently with the following capabilities:

- **High concurrency**

  The platform can handle multiple concurrent queries without significant performance degradation, making it ideal for organizations with many BI users.

- **Complex query optimization**

  Advanced query optimization techniques ensure that even complex analytical queries run efficiently.

- **Consistent performance**

  The combination of caching and serverless architecture provides consistent query performance, even as data volumes grow.

Databricks SQL Serverless is the best choice for BI workloads due to its innovative approach to data processing and storage. The disk cache feature significantly enhances query performance, while the serverless architecture provides unparalleled flexibility and scalability. By decoupling storage from compute and offering a pay-for-use model, it provides a cost-effective solution that can adapt to the changing needs of modern businesses.

Databricks SQL Serverless offers a powerful, efficient and scalable platform that can handle the most demanding analytical workloads.

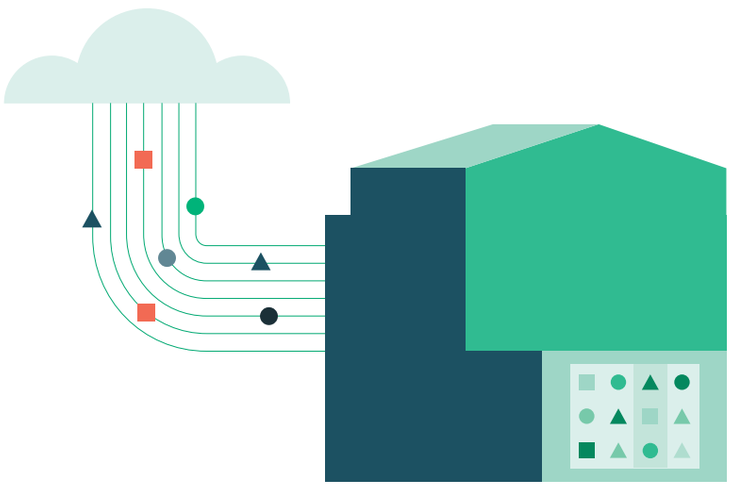
CHAPTER

02

Use Cases

Building data applications

The ability to create powerful, user-friendly data applications has become a critical skill for data and AI teams. This chapter explores the concept of building production data applications, focusing on how data professionals can leverage their existing skills in Python and SQL to create sophisticated, interactive tools that bridge the gap between complex data analysis and actionable insights for nontechnical users.



Key components for building data applications

Low-code or no-code application builders

For line of business (LOB) users to create their own applications, they need a low-code or no-code application building solution which removes the need to learn new front-end development technologies. This way, data professionals can focus on their strengths while delivering robust, interactive applications.

Databricks Platform

Databricks solves the challenge of data unification, governance and AI integration. By natively embedding AI into data with Unity Catalog and Databricks Assistant, developers can create applications that seamlessly leverage the intelligence within their environment, including models, assistants and Unity Catalog context.

Unity Catalog

Unity Catalog plays a crucial role in governing the data, functions and AI models used in data applications. It allows for centralized management of data access and provides a unified view of metadata across the Databricks Data Intelligence Platform.

Use case examples

To illustrate the power and flexibility of building data applications on Databricks, let's explore two practical use cases:

Tagging policy management

This application addresses the common challenge of ensuring proper usage tagging across an organization's Databricks environment. Key features include:

- Automatic schema, table and ETL deployment using materialized views
- Parameterized SQL statements for tagging policy adherence analysis
- Write-back tables for user-defined tags, policies and annotations
- A user-friendly interface for nontechnical users to perform analysis and make changes

The application allows users to:

- Define tag policies
- Analyze adherence to these policies across the organization
- Identify and fix noncompliant usage directly within the app
- Visualize compliance trends over time

LLM-generated alerts on system tables

This application simplifies the process of creating and managing alerts based on system table data. It features:

- A SQL function registered in Unity Catalog that utilizes AI_QUERY for natural language processing
- A chat interface for users to interact with the model and create alerts
- Write-back functionality to register queries, create alerts and schedule Databricks Jobs
- A management interface for editing, removing and viewing saved alerts

With this application, users can:

- Submit natural language queries to generate alerts
- Refine and customize alert parameters through conversation with the AI
- Save and schedule alerts automatically
- Manage all created alerts in one centralized location

Building blocks of data applications

Callbacks and state management

Your low-code or no-code solution needs to handle user inputs, trigger data processing and update the UI accordingly. For example, the tag policy management app uses callbacks to handle policy changes, save data to write-back tables and update visualizations in real time.

SQL integration

Leveraging Databricks SQL, applications can execute complex queries directly from Python code. This allows for seamless integration of data processing and visualization within the same application.

AI integration

By incorporating AI_QUERY functions and the Databricks Assistant, applications can provide natural language interfaces for complex data tasks, as demonstrated in the alert generation use case.

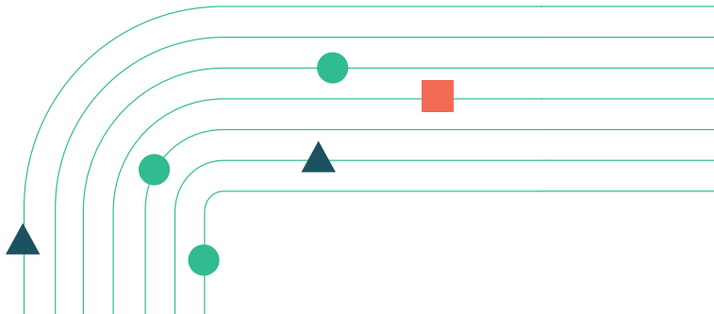
Write-back capabilities

The ability to write data back to tables in Unity Catalog allows for the creation of truly interactive applications where users can not only view but also modify and create data.

Job scheduling and alerting

Integration with Databricks Jobs enables applications to schedule and manage recurring tasks, such as running alerts or updating dashboards.

Building data applications on Databricks represents a paradigm shift in how data and AI teams can deliver value to their organizations. Data professionals can now create sophisticated, interactive applications that put the power of data directly into the hands of business users. The ability to build these types of applications will become an increasingly valuable skill for data and AI teams, allowing them to scale their impact and drive innovation within their organizations.



databricks

Sensitive data governance design patterns in Databricks SQL

In the era of big data and stringent privacy regulations, sensitive data governance has become a critical aspect of data management. This section explores various design patterns for implementing sensitive data governance in Databricks SQL, focusing on how organizations can effectively control access to sensitive information while maintaining data utility.

The foundation of sensitive data governance in Databricks is Unity Catalog, which serves as the backbone for unified governance and visibility across all data assets. There are four primary design patterns: views, data masking, dynamic views and row-level security and column masking. Each pattern offers unique advantages and use cases for managing sensitive data access.

Views

Views are the most straightforward and commonly used abstraction pattern for controlling data access. They provide a read-only interface to underlying data, allowing you to expose only nonsensitive columns or sanitized versions of sensitive data.

Key considerations

- Explicitly specify columns instead of using SELECT * to avoid unintentionally exposing new sensitive columns
- Maintain simplicity in view definitions to ensure ease of understanding and debugging

```
CREATE VIEW sales_redacted AS
SELECT
  user_id,
  CASE WHEN
    is_group_member('auditors') THEN email
    ELSE 'REDACTED'
  END AS email,
  country,
  product,
  total
FROM sales_raw
```

Data masking

Data masking protects sensitive parts of specific columns by applying masking functions at query time. This approach is useful when you need to retain some utility of the data while obscuring sensitive details.

Key considerations

- Prioritize removing sensitive data over masking when possible
- Utilize AI-based masking (ai_mask function) for complex scenarios with mixed data types
- Consider encryption for highly sensitive data that requires privileged access

```
CREATE VIEW sales_redacted AS
SELECT
  user_id,
  region,
  CASE
    WHEN is_account_group_member('auditors'
    ELSE regexp_extract(email, '^.*@(.*)$',
  END
  FROM sales_raw
```

Dynamic views

Dynamic views offer more sophisticated access control by applying filters or masking based on user groups at runtime. They're particularly useful when different user groups require varying levels of access to the same dataset.

Key considerations

- Implement group membership–based checks rather than user-specific checks
- Use dynamic views when maintaining separate views per group becomes cumbersome
- Consider modularizing filters into user-defined functions (UDFs) for better maintainability

```
USE CATALOG `_demo_catalog`;
USE DATABASE staging;

CREATE OR REPLACE FUNCTION redact_income (annual_inc FLOAT)
RETURN CASE WHEN
-- If the user belongs to the specified account group let them see the annual_income column data
  is_account_group_member('data science team') THEN annual_inc
-- Otherwise replace the annual_income column values with the following specified string
  ELSE 'redacted'
END;

CREATE OR REPLACE FUNCTION ownership_filter(home_ownership STRING)
RETURN CASE WHEN
  is_account_group_member('data science team') THEN TRUE
  ELSE home_ownership = "MORTGAGE"
END;
```

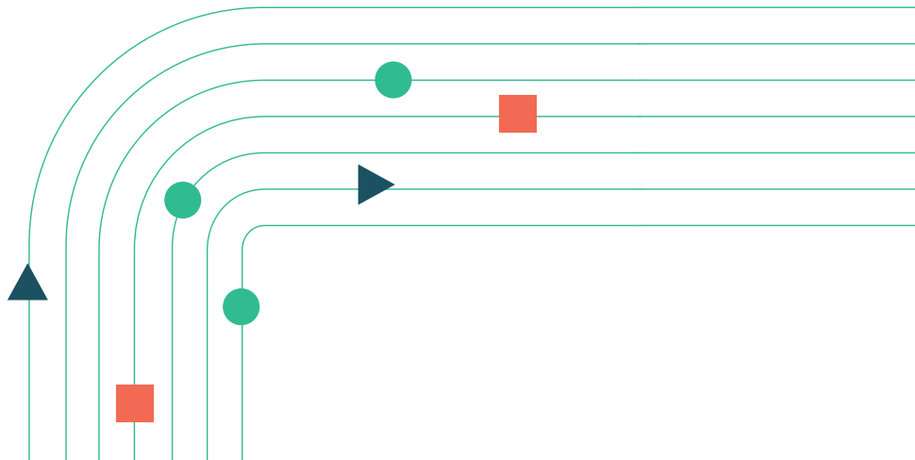
Row filters and column masking

Row-level security (RLS) and column-level masking (CLM) are advanced features that allow fine-grained access control directly at the table level in Unity Catalog. These can be applied during table creation or later using ALTER TABLE statements.

Key considerations

- Use mapping tables for complex security models based on organizational structure
- Store SQL user-defined functions (UDFs) in Unity Catalog for consistent application across all table access operations
- Be aware of limitations associated with tables using row filters and column masks

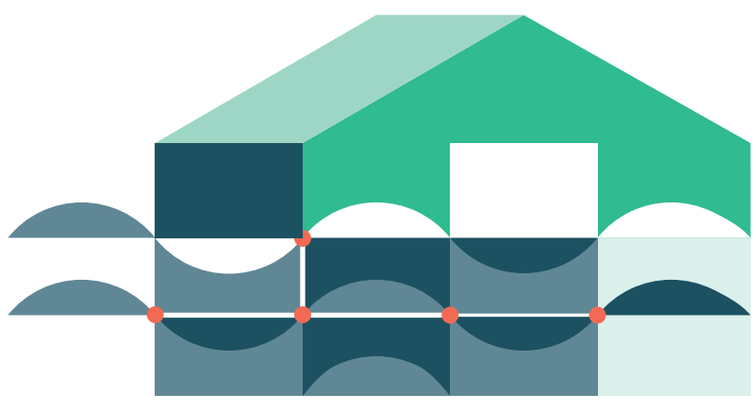
```
ALTER TABLE loan_data ALTER COLUMN annual_inc SET MASK redact_income;
ALTER TABLE loan_data SET ROW FILTER ownership_filter ON
(home_ownership);
```



Best practices and recommendations

- **Data minimization:** Where possible, remove sensitive data instead of masking or filtering.
- **Simplicity:** Use the simplest pattern that meets your requirements. Start with regular views and progress to more complex patterns only when necessary.
- **Group-based access:** Implement access controls based on group membership rather than individual users for better scalability and management.
- **Modularization:** Break complex logic into reusable components such as UDFs or lookup tables to improve maintainability.
- **Combine patterns:** These patterns aren't mutually exclusive. Combine them as needed to achieve the desired level of data protection and access control.
- **Regular audits:** Periodically review and audit your sensitive data governance implementations to ensure they align with evolving business needs and compliance requirements.
- **Documentation:** Maintain clear documentation of your data governance patterns, including the reasoning behind each implementation, to facilitate future maintenance and knowledge transfer.

Effective sensitive data governance is crucial for maintaining data security, compliance and trust. By leveraging the design patterns discussed in this section — views, data masking, dynamic views, row-level security and column masking — organizations can implement robust and flexible data access controls in Databricks SQL.



Dynamic SQL design patterns with Databricks SQL on workflows

This section explores advanced dynamic SQL design patterns that leverage the latest features of Databricks SQL, including SQL variables, EXECUTE IMMEDIATE and the ability to run SQL notebooks on Databricks SQL with Databricks Workflows. These capabilities enable data engineers and analysts to create sophisticated, flexible and maintainable data pipelines entirely within the SQL ecosystem.

Key components of dynamic SQL design patterns

SQL variables

SQL variables allow for dynamic value assignment and manipulation within SQL scripts. They enable more flexible and parameterized queries, reducing the need for hard-coded values and improving maintainability.

EXECUTE IMMEDIATE

This feature allows for the execution of dynamically constructed SQL strings, opening up possibilities for adaptive query execution based on runtime conditions.

SQL notebooks on Databricks SQL with workflows

The ability to run SQL notebooks on Databricks SQL within workflows brings the power of notebook-based development to pure SQL environments, enabling version control, parameterization and seamless orchestration of complex SQL pipelines.

Design pattern: Incremental ETL with change data feed

One powerful design pattern enabled by these features is the implementation of incremental ETL pipelines using change data feed (CDF). This pattern allows for efficient processing of only the changed data between runs, reducing processing time and resource usage.

Key steps in this pattern include the following.

Dynamic DDL generation

Use SQL variables and EXECUTE IMMEDIATE to dynamically create or alter table structures based on runtime conditions.

```
DECLARE OR REPLACE sqlStr = 'SELECT AVG(c1) FROM VALUES(?), (?) AS t(c1)';
DECLARE OR REPLACE arg1 = 5;
DECLARE OR REPLACE arg2 = 6;
EXECUTE IMMEDIATE sqlStr USING arg1, arg2;
```

Version tracking with checkpoint tables

Implement checkpoint tables to track the last processed version of source tables, enabling efficient incremental processing.

```
SELECT CONCAT('Latest Version Processed: ', checkpoint_version_bronze_to_silver::string) AS latest_check,
CONCAT('Next Checkpoint Max Version For Active Batch: ', next_temp_checkpoint::string) AS next_check;
```

Dynamic version retrieval

Use SQL variables to dynamically retrieve the last processed version and determine the next version to process.

```
CREATE TABLE versions (software STRING, version STRING)  TBLPROPERTIES (delta.enableChangeDataFeed = true);
INSERT INTO versions VALUES ('IDE', '6.2.0');
UPDATE versions SET version = '5.1.0' WHERE software = 'IDE';
INSERT INTO versions VALUES ('IDE-1', '1.3.0');
DELETE FROM versions WHERE version = '1.3.0';
SELECT max(_commit_version), max(version) FROM table_changes('versions', 2);
```

Incremental MERGE operations

Implement MERGE statements that use the dynamically determined version range to process only the changed data.

```
USING (SELECT * FROM source WHERE created_at >= (current_date() - INTERVAL '5' DAY)) AS s
ON t.key = s.key
WHEN MATCHED THEN UPDATE SET *
WHEN NOT MATCHED THEN INSERT *
WHEN NOT MATCHED BY SOURCE AND created_at >= (current_date() - INTERVAL '5' DAY) THEN DELETE
```

Checkpoint update

After successful processing, update the checkpoint table with the new latest version.

```
SELECT CONCAT('Latest Version Processed: ', checkpoint_version_bronze_to_silver::string) AS latest_check,
CONCAT('Next Checkpoint Max Version For Active Batch: ', next_temp_checkpoint::string) AS next_check;
```

Design pattern: Dynamic view management

Another powerful pattern is the dynamic management of views, enabling scenarios like blue/green deployments or conditional view definitions based on data quality checks.

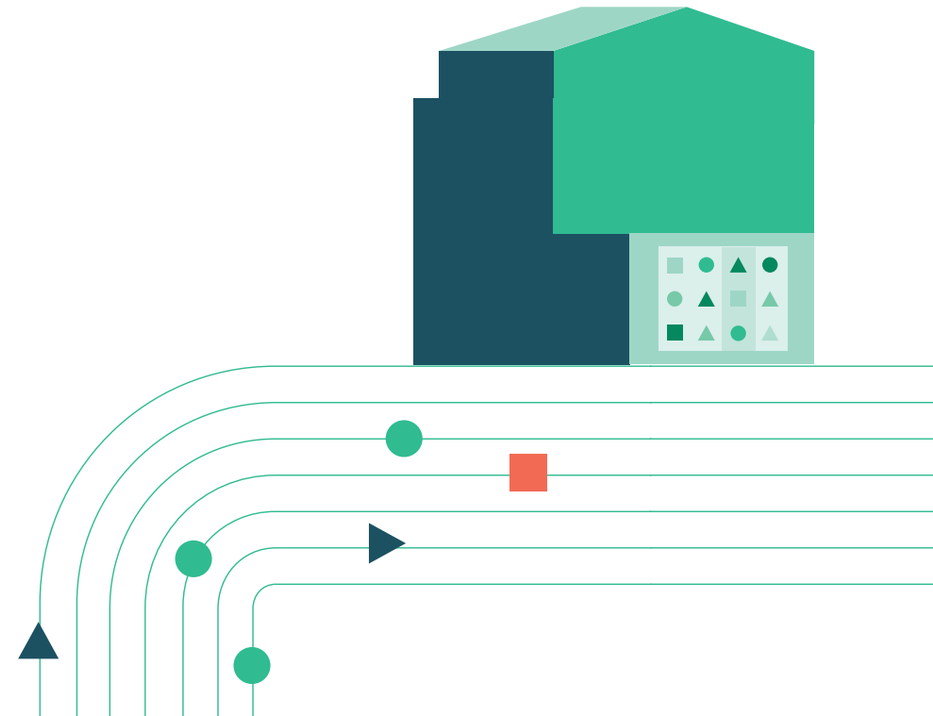
Key steps in this pattern include the following.

Version selection based on data quality rules

Use SQL variables and conditional logic to determine which version of the underlying tables should be used in the view definition.

Dynamic view creation

Use EXECUTE IMMEDIATE to dynamically create or replace views based on the selected version.



Orchestration with Databricks Workflows

These dynamic SQL patterns can be encapsulated within SQL notebooks and orchestrated using Databricks Workflows. This approach provides several benefits.

Parameterization
Job and task-level parameters can be referenced in SQL notebooks, allowing for flexible, parameterized workflows.

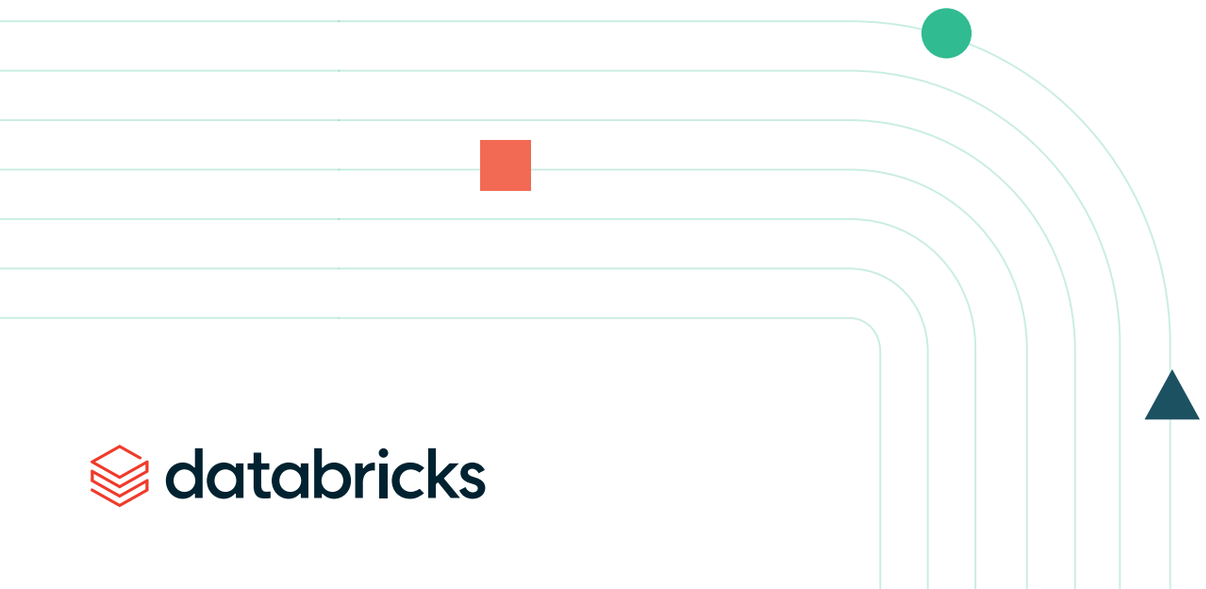
Version control
SQL notebooks can be version controlled, enabling better collaboration and change management.

Observability
Databricks Workflows provide a unified interface for monitoring and managing the execution of complex SQL pipelines.

Best practices and recommendations

- **Modularization:** Break down complex ETL logic into separate SQL notebooks for better maintainability.
- **Error handling:** Implement robust error handling and logging within your SQL scripts to facilitate troubleshooting.
- **Performance optimization:** Leverage Databricks features like Liquid Clustering and CDF to optimize query performance.
- **Testing:** Implement unit tests for individual SQL components and integration tests for the entire pipeline.
- **Documentation:** Provide clear inline comments and external documentation to explain complex logic and design decisions.

Dynamic SQL design patterns in Databricks SQL, coupled with the ability to run SQL notebooks on Databricks SQL with Databricks Workflows, represent a significant advancement in data warehousing capabilities. These patterns enable data and AI teams to create sophisticated, maintainable and efficient ETL pipelines entirely within the SQL ecosystem. By leveraging SQL variables, EXECUTE IMMEDIATE and CDF, data engineers can implement incremental processing, dynamic view management and other advanced patterns that were previously challenging or impossible with traditional SQL alone.



End-to-end streaming ELT on Databricks SQL with streaming tables and materialized views

The landscape of data processing has evolved significantly, with real-time data becoming increasingly crucial for businesses. Databricks SQL includes two powerful features — streaming tables (STs) and materialized views (MVs) — that enable data and AI teams to implement end-to-end streaming extract, load and transform (ELT) pipelines directly within the SQL environment. This section explores how these features can be leveraged to create a robust, real-time medallion architecture, unifying batch and streaming capabilities within a single engine.

Understanding the medallion architecture

As noted earlier in this guide, the lakehouse medallion architecture is a data modeling approach that organizes data into Bronze, Silver and Gold quality tiers.

This architecture provides a clear path for data refinement and enables different teams to work with data at the appropriate level of quality and aggregation.

Streaming tables and materialized views

Streaming tables

- Unity Catalog-managed tables supporting append-only incremental and streaming data processing
- Automatically create and manage Delta Live Tables (DLT) pipelines behind the scenes
- Ideal for Bronze and Silver layer tables in the medallion architecture

Materialized views

- Unity Catalog-managed tables storing precomputed results based on the latest version of source data
- Results reflect the state of data at the last refresh, not necessarily real time
- Beneficial for improving query performance and reducing costs, particularly useful for the Gold layer

Governance and maintenance

Lineage and governance

Utilize Unity Catalog for automatic lineage tracking and governance. You can visualize the entire pipeline's lineage, including upstream and downstream dependencies.

Refreshing materialized views

Set up refresh queries for materialized views:

```
CREATE OR REFRESH MATERIALIZED VIEW churn_orders_bronze
COMMENT “Spending score from raw data”
AS SELECT *  FROM read_files(
    ‘/Volumes/mail/pearl_ubaru_retail/retail/orders’,
    format => ‘json’,
    header => true,
    mode => ‘FAILFAST’)
```

Monitoring refresh operations

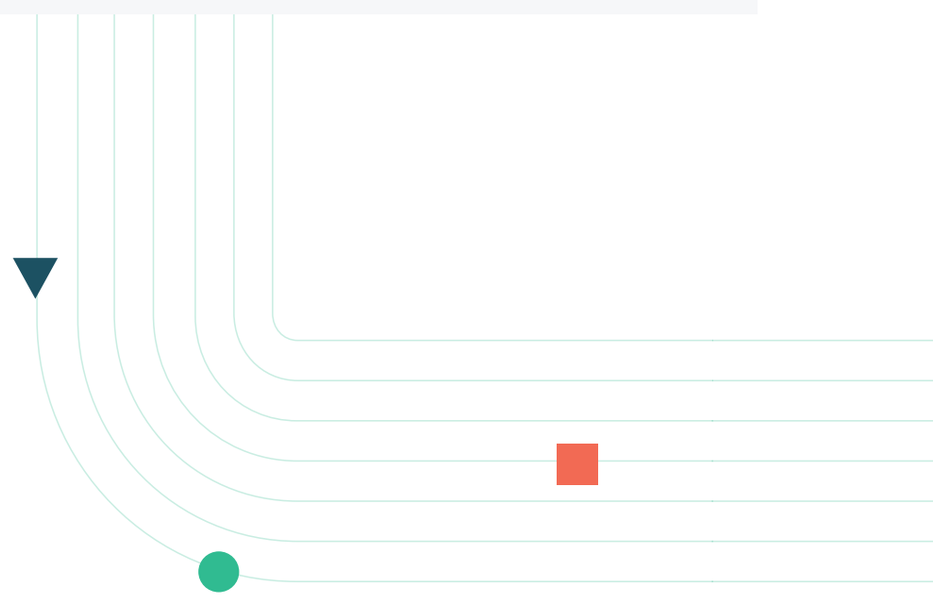
Query the event log to monitor refresh operations:

```
SELECT
  *
FROM
  event_log(TABLE(churns_orders_silver))
WHERE
  event_type = "update_progress"
ORDER BY
  timestamp desc;
```

Scheduling automatic refreshes

Schedule automatic refreshes for streaming tables and materialized views:

```
ALTER MATERIALIZED VIEW churns_orders_bronze
      ADD SCHEDULE CRON '0 0 0 * * ? *' AT TIME ZONE 'America/Los_Angeles';
```



Best practices and considerations

- Choose wisely between streaming tables and materialized views:
  - Use streaming tables for processing data streams as they arrive
  - Use materialized views for frequently accessed or commonly queried data
- Implement data quality checks early in the pipeline to ensure data integrity.
- Utilize Z-Ordering for materialized views that are frequently queried to improve performance.
- Regularly monitor and analyze the quarantine table to improve data quality at the source.
- Leverage Unity Catalog's lineage capabilities for data governance and impact analysis.
- Optimize refresh schedules based on data update frequency and business requirements.

The combination of streaming tables and materialized views in Databricks SQL provides a powerful toolkit for implementing end-to-end streaming ELT pipelines. By leveraging these features within the medallion architecture, data and AI teams can create robust, real-time data processing workflows entirely within the SQL environment. This approach not only simplifies the architecture by unifying batch and streaming processes but also enhances data governance through Unity Catalog integration. As data needs continue to evolve, this Databricks SQL-centric approach offers a flexible, scalable solution for modern data engineering challenges.

databricks

Enabling data analysts with dashboards on your data warehouse

Enabling data analysts to efficiently extract insights from data warehouses is crucial for business success. This section explores the evolution of BI tools, focusing on the latest advancements in AI-powered dashboards and conversational interfaces. These tools are transforming the way analysts interact with data, and Databricks AI/BI is a prime example of this new generation of analytics platforms.

The evolution of BI

Traditionally, business users have relied on static reports and dashboards to answer their data questions. However, this approach has several limitations:

- **Inflexibility:** Pre-built dashboards can't anticipate all possible questions
- **Dependence on data and AI teams:** New visualizations often require assistance from overworked data professionals
- **Time lag:** The cycle of requesting and creating new dashboards can be slow and inefficient

Databricks AI/BI

Databricks AI/BI consists of two main components:

- AI/BI dashboards
- Genie (conversational interface)

Key features of AI/BI dashboards

- **AI-powered, low-code authoring:** Simplifies the process of creating dashboards
- **Standard BI capabilities:** Includes visualizations, cross-filtering and scheduled reporting
- **Integration with data platform:** No need for separate semantic models or data extracts
- **Seamless exploration:** Easy transition to Genie for deeper analysis

Conversational analytics using Genie

Genie extends beyond traditional dashboards by offering:

- **Natural language interface:** Users can ask questions in plain language
- **Flexible analysis:** Not limited to predefined charts or metrics
- **Clarification and suggestion:** Asks for clarification when needed and proposes alternative analyses
- **Learning capability:** Improves over time based on user interactions and feedback

The compound AI system

At the core of AI/BI is a compound AI system that leverages multiple AI agents, each specializing in specific tasks:

- Planning
- SQL generation
- Explanation
- Visualization
- Result certification

This ensemble approach allows for more robust and accurate analysis compared to single-model systems.

Ensuring trust and accuracy

AI/BI addresses the critical issue of trust in AI-generated analytics through several mechanisms:

- **Transparency:** The system explains its reasoning and sources
- **Human feedback loop:** Analysts can verify assumptions and provide corrections
- **Certified answers:** Integration with trusted, governed logic (e.g., Unity Catalog functions and metrics)
- **Quality monitoring:** Data and AI teams can tune and benchmark the system's performance

Platform integration benefits

By integrating deeply with the underlying data platform, AI/BI offers several advantages:

- **Unified governance and lineage:** Leverages existing data governance frameworks
- **Effortless sharing:** Easy to share analyses without additional user licenses
- **High performance:** Utilizes optimized data warehouse and query engine capabilities
- **Data freshness:** No need for separate data extracts, ensuring up-to-date analysis

Empowering analysts and business users

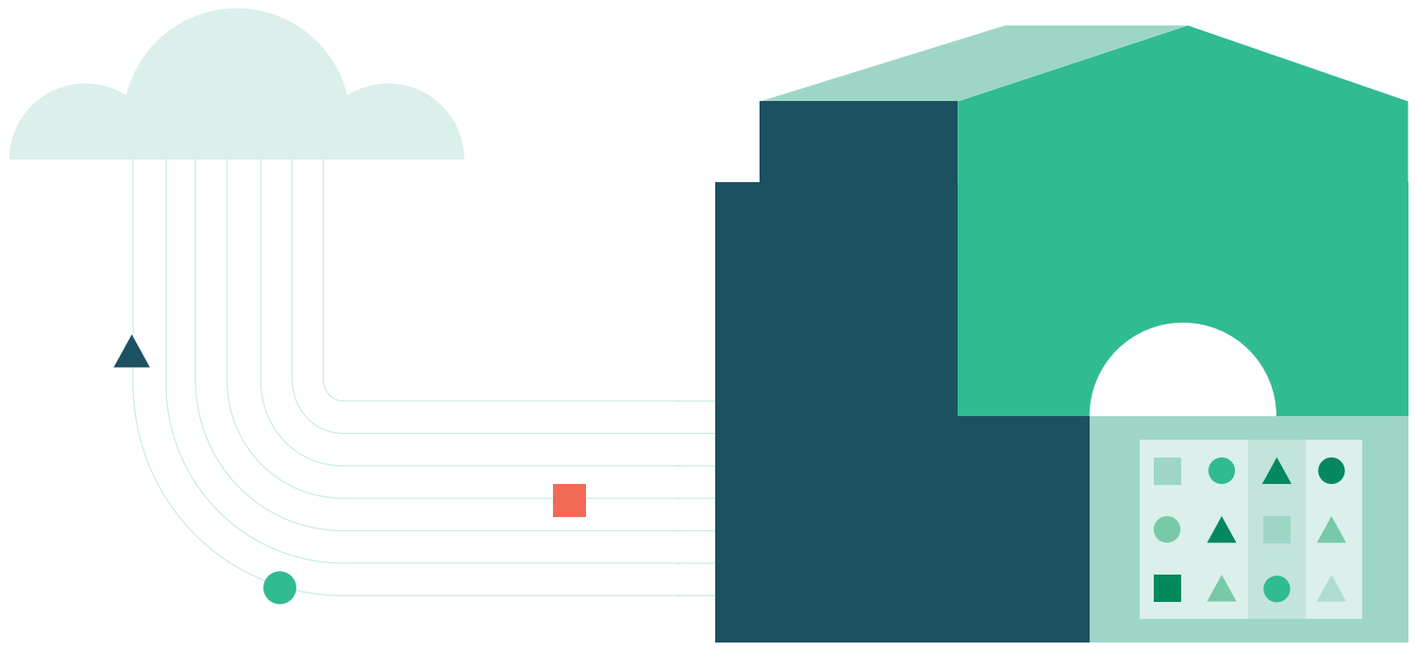
The combination of AI-powered dashboards and conversational interfaces significantly empowers data analysts and business users:

- **Self-service analytics:** Users can answer a broader range of questions independently
- **Reduced dependence on data and AI teams:** Frees up data professionals for more complex tasks
- **Faster insights:** Shortens the time from question to answer
- **Continuous learning:** The system improves with use, becoming more valuable over time

Best practices for implementation
When implementing AI-powered BI tools like Databricks AI/BI, consider the following best practices:

- **Data quality:** Ensure your data warehouse contains clean, well-structured data.
- **User training:** Provide guidance on how to effectively interact with the AI system.
- **Feedback culture:** Encourage users to provide feedback to improve the system's accuracy.
- **Governance framework:** Establish clear guidelines for data access and usage.
- **Performance monitoring:** Regularly assess the system's accuracy and efficiency.

The introduction of AI-powered dashboards and conversational interfaces represents a significant leap forward in enabling data analysts to extract value from data warehouses. By combining the strengths of traditional BI dashboards with the flexibility and power of AI-driven conversational interfaces, Databricks SQL is democratizing data analysis and accelerating the pace of data-driven decision-making.



CHAPTER

03

End-to-End Steps

In the ever-evolving landscape of data management, platform owners are tasked with architecting robust pipelines that seamlessly integrate, process and analyze data. This guide unfolds the journey of leveraging Databricks SQL to build a comprehensive data warehousing solution, highlighting key stages and capabilities.

Ingest and transform

Data sources and connectors

The journey begins with Lakeflow Connect, a powerful data ingestion tool within Databricks SQL. Lakeflow Connect integrates diverse data sources into the lakehouse, supporting a wide array of sources including AWS S3, Azure Blob Storage, Google Cloud Storage, MySQL, PostgreSQL, SQL Server, Kafka, Kinesis and Event Hubs.

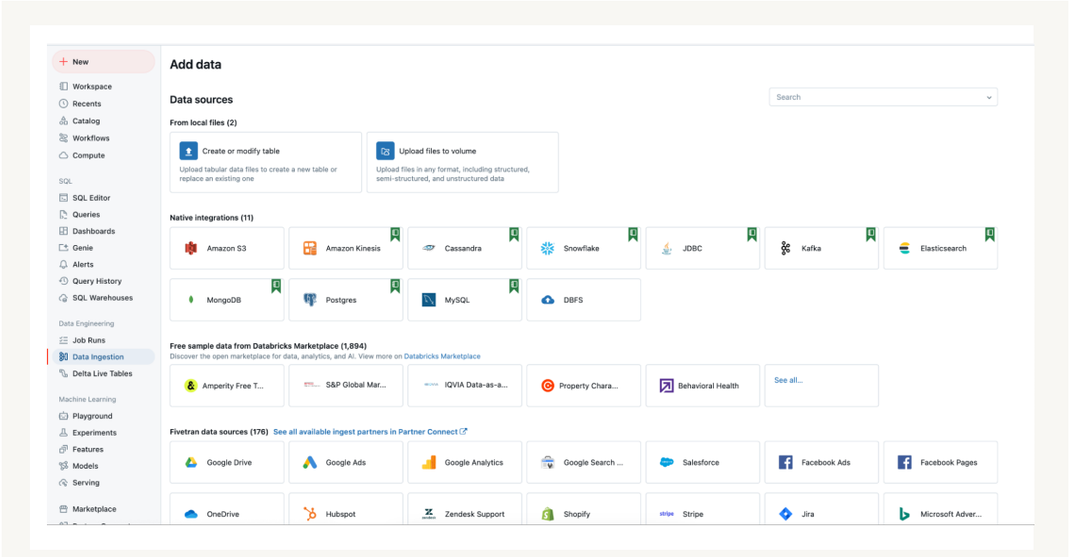
Databricks also offers Partner Connect to enhance the functionality of your lakehouse environment. Partner Connect enables you to seamlessly integrate a wide array of data, analytics and AI solutions directly within the Databricks Platform, streamlining the process of discovering and incorporating new tools. For organizations looking to extend their data ingestion and transformation capabilities beyond Databricks native offerings, Partner Connect provides access to over 600 connectors from industry-leading providers such as Fivetran, Qlik, Informatica, dbt and Alteryx. This extensive ecosystem empowers users to efficiently manage their data workflows with just a few clicks.

Lakeflow Connect offers flexible ingestion methods to suit your needs:

- **Batch ingestion:** Schedule batch ingestions from cloud files, databases and other sources to run at regular intervals
- **Streaming ingestion:** Process and analyze real-time data from sources like Kafka, Kinesis and Event Hubs
- **Change data capture (CDC):** Capture changes made to data in real time, ensuring the lakehouse is always in sync with the source data

Once data is ingested, Lakeflow Connect automatically handles data processing, including validation, cleansing and transformation. The data is then stored in the lakehouse, ready to be queried and analyzed using Databricks SQL.

By simplifying data ingestion, Lakeflow Connect reduces latency and increases data freshness, enabling you to make better decisions with your data.



Tools and transformations

Now that the data source integration is established, the next step is to ensure the ease of pipeline creation and maintenance. Databricks enables the creation of comprehensive data pipelines with Delta Live Tables (DLT) and dbt, offering a low-code/no-code solution for efficient pipeline creation. With DLT, you can build and schedule complex workflows, known as directed acyclic graphs (DAGs), to manage both batch and streaming processes. This allows for incremental or complete data refreshes, ensuring your data remains up-to-date and relevant.

DLT provides a range of capabilities to simplify data transformation, including:

- **Event-driven triggers:** Initiate workflows based on specific events, such as data arrival or changes to metadata
- **Scheduled tasks:** Schedule workflows to run at specific times or intervals
- **Data quality expectations:** Define and enforce data quality rules to ensure data accuracy and consistency
- **Auto-incrementing identity columns:** Easily manage unique identifiers for your data
- **Slowly changing dimensions (SCD) types 1 and 2:** Capture changes to dimension tables with ease, eliminating the need for complex auxiliary computations

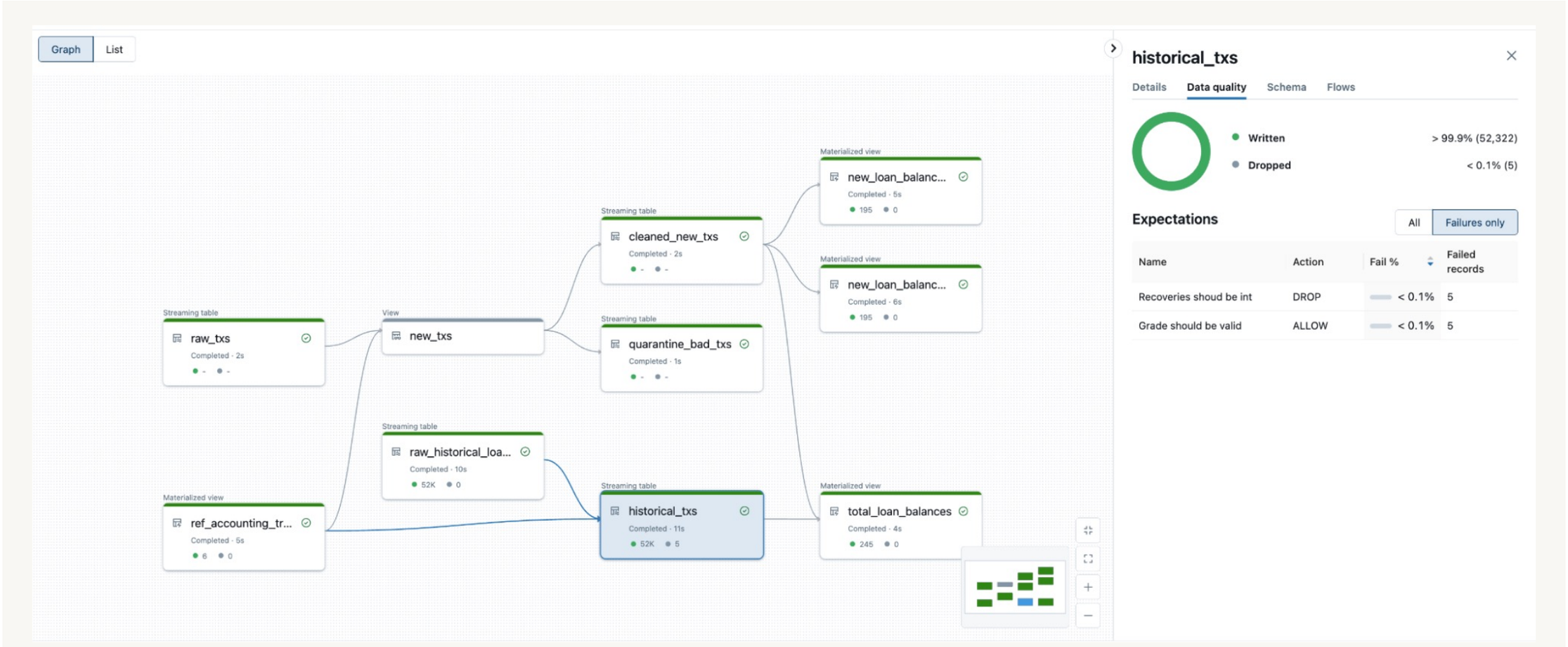
DLT also captures quality statistics and I/O statistics, with observability dashboards to monitor pipeline performance and data quality. This helps identify areas for optimization and ensures your data meets the highest standards.

Creating a DLT pipeline can be as simple as writing a SQL query, with no extensive coding knowledge required. For example:

```
CREATE MATERIALIZED VIEW my_silver_table
SCHEDULE CRON '0 0 * ? * * *'
AS SELECT count(distinct event_id) as event_count from my_bronze_table;
```

This low-code approach enables data engineers and analysts to focus on the logic of the pipeline, without worrying about the underlying infrastructure. DLT also supports a range of programming languages, including SQL, Python, R and Scala, allowing them to work with the language of their choice.

Databricks SQL and DLT can revolutionize data transformation processes, prioritizing data quality and simplifying historical data management. Engineers and analysts can create efficient, scalable and reliable data pipelines that meet the needs of their organization while minimizing the need for extensive coding knowledge.



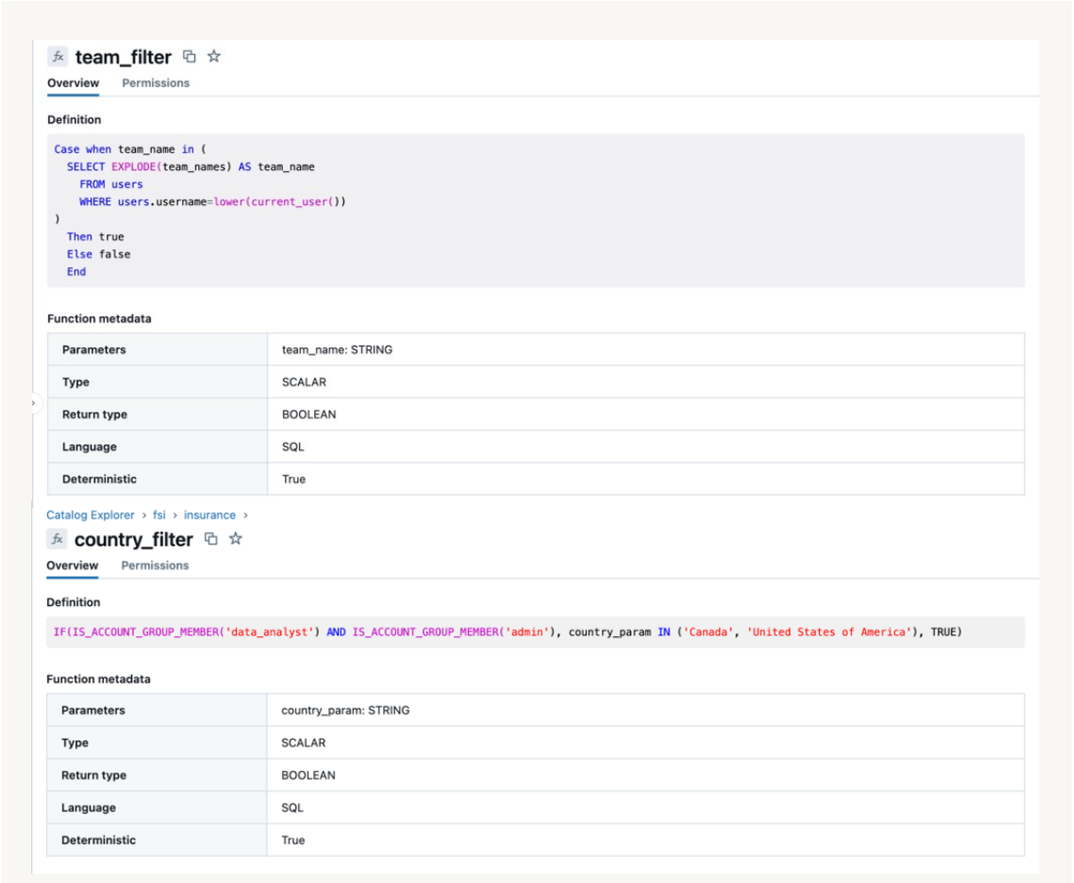
Governance and traditional data warehouse capabilities

The data is now available after ingestion using low code pipelines at various intervals and event triggers from multiple data source integrations, but how does one govern this data? Data governance is the foundation upon which data-driven organizations are built, fostering a culture of transparency, accountability and innovation. Databricks robust governance and security allow teams to:

- Secure data mart schemas and warehouse schemas using Unity Catalog's role-based access control (RBAC) with grant, revoke and deny capabilities. This ensures that only authorized users have access to sensitive data.
- Apply fine-grained privacy control measures with row-level security (RLS) to restrict access to sensitive data. Define row-level security policies to control access to specific rows in a table based on user identity and privileges.
- Use column masks to dynamically mask sensitive columns based on user identity and privileges. Create a column mask that reveals sensitive data only to authorized users, while masking it for unauthorized users. Define a column mask that uses a runtime function to determine the user's identity and privileges, and apply it to sensitive columns.

To create a column mask that reveals sensitive data only to authorized users, follow these steps:

- Define a runtime function that checks the user's identity and privileges
- Create a column mask that uses the runtime function to determine whether to reveal or mask the sensitive data
- Apply the column mask to the sensitive column



Unity Catalog provides a centralized approach to data governance, enabling:

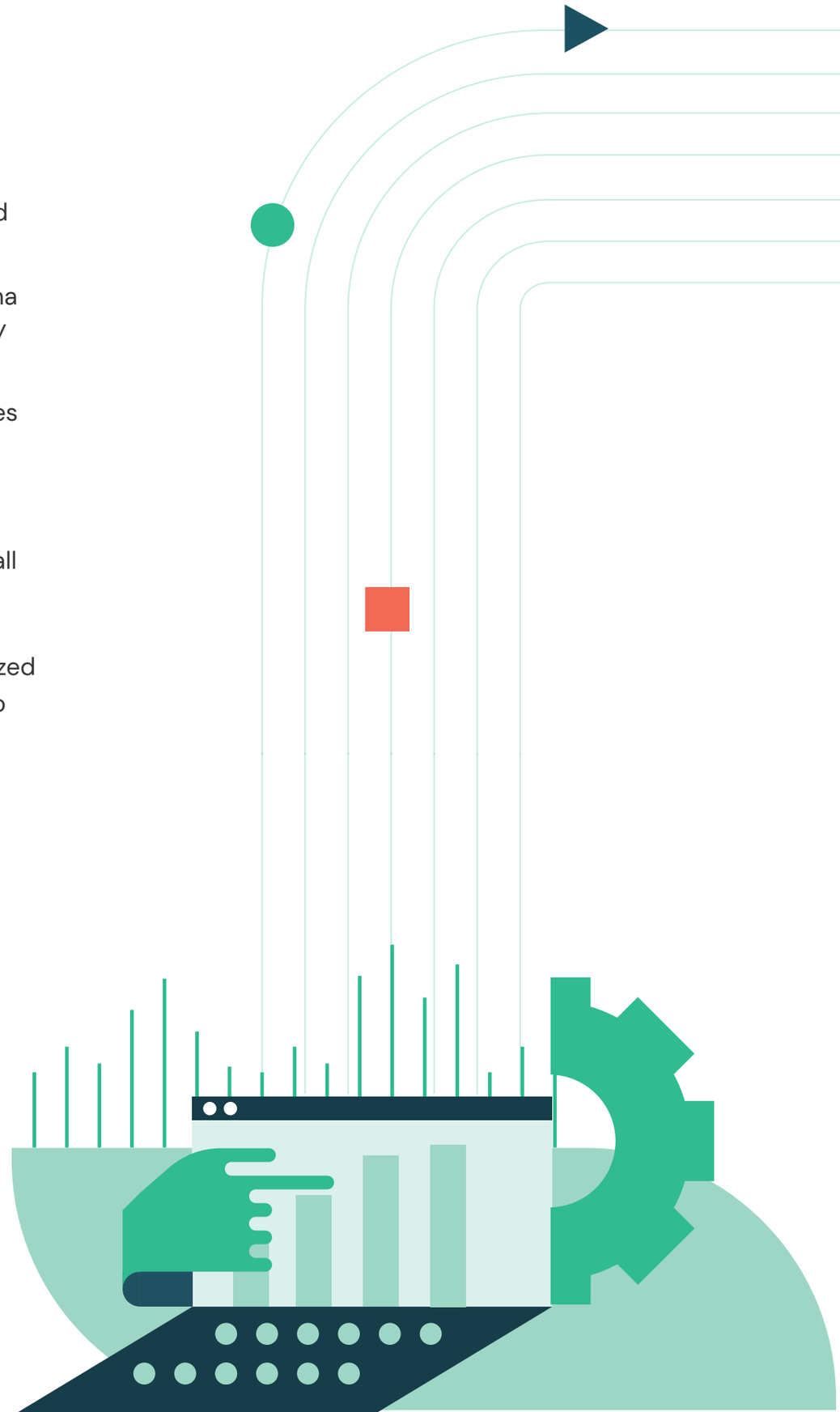
- **Automatic lineage building:** Track data origins to ensure data accuracy and consistency at table and column levels
- **Automatic documentation:** Generate documentation to provide context and meaning to the data
- **Simplified metadata management:** Centrally manage metadata to ensure data consistency and accuracy
- **Security controls**: Apply security controls to restrict access to sensitive data

Delta Lake ensures data consistency and integrity with:

- **ACID transactions:** Ensure reliable and secure processing
- **Schema enforcement:** Enforce schema constraints to ensure data consistency and accuracy
- **Time travel capabilities:** Track changes over time for data accuracy and consistency
- **Delta transaction log:** Provide a transparent and immutable record of all data changes

Additionally, Delta Lake supports materialized views for optimized query performance. To use materialized views, follow these steps:

- Create a materialized view to cache frequently accessed data
- Use the materialized view to optimize query performance



Enable AI-powered SQL querying

Databricks offers a comprehensive suite of querying tools and features to efficiently explore, analyze and derive insights from their data. This section of our guide focuses on the querying capabilities within the Databricks Platform.

SQL editor: The heart of data exploration
The Databricks SQL editor is a powerful interface for writing and executing SQL queries. It provides a user-friendly environment with the following features designed to enhance productivity and data discovery:

- **Autocomplete and syntax highlighting:** Intelligently suggests table names, columns and SQL keywords as you type
- **Schema browser:** Easily explore available databases, tables and columns directly within the editor
- **Query history:** Access and reuse previously executed queries for efficient workflow
- **Visualization tools:** Create charts and graphs directly from query results for quick data insights

AI-powered data discovery and query assistance
Databricks leverages artificial intelligence to revolutionize the querying process:

- **Databricks Assistant:** An AI-powered tool that generates SQL queries based on natural language input, making data exploration accessible to users of all skill levels
- **AI snippet generation:** Automatically suggests and generates code snippets to accelerate query writing
- **Databricks Clean Rooms:** Secure environments for collaborative data analysis, ensuring data privacy and compliance
- **Genie spaces:** Data analysts configure Genie spaces with datasets, sample queries and text guidelines to help Genie translate business questions into analytical queries

Lakehouse Federation: Unified query access
Lakehouse Federation enables seamless querying across multiple data sources:

- **Query federation:** Run queries against external data sources without data migration
- **Single pane of glass:** Access and analyze data from various sources through a unified interface
- **Simplified data discovery:** Explore and query data across your entire data ecosystem effortlessly

AI functions in Databricks SQL

Databricks SQL incorporates AI functions directly into the querying process, enabling these advanced analytics capabilities without complex ML pipelines:

- **Sentiment analysis:**
  Analyze text data for sentiment; example:

```
SELECT text, AI_SENTIMENT(text) AS sentiment
FROM customer_feedback;
```

- **Language translation:**
  Translate text columns on the fly during queries; example:

```
SELECT product_name, AI_TRANSLATE(description, 'EN', 'ES') AS
spanish_description FROM products;
```

- **Text summarization:**
  Generate concise summaries of large text fields; example:

```
SELECT article_title, AI_SUMMARIZE(article_content, 100) AS
summary FROM news_articles;
```

- **Named entity recognition:**
  Extract entities from text data; example:

```
SELECT document_id, AI_EXTRACT_ENTITIES(document_text) AS
entities FROM legal_documents;
```

These AI functions enable advanced analytics capabilities directly within SQL, democratizing access to AI-powered insights.

Natural language querying and assisted query building

Databricks natural language querying bridges the gap between business users and complex data structures with the following capabilities:

- **Conversational interface:** Ask questions about your data in plain English
- **Query translation:** Automatically convert natural language questions into optimized SQL queries
- **Iterative refinement:** Engage in a dialogue to refine and improve query results

Query optimization and debugging

Databricks provides tools to ensure optimal query performance:

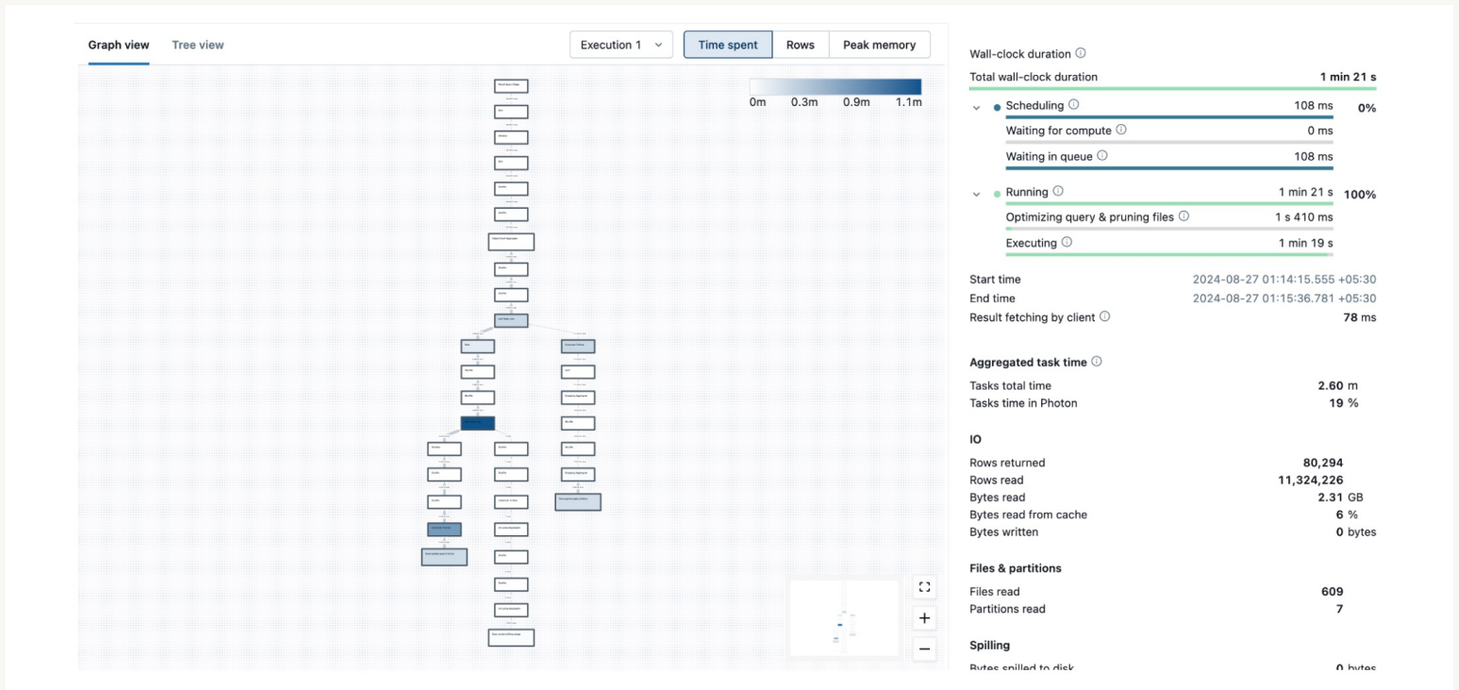
- **Query Profile:** Visualize query execution plans and identify bottlenecks
- **AI-assisted optimization:** Receive intelligent suggestions for query improvements
- **Automated error detection:** Quickly identify and resolve syntax errors and logical issues in queries

Code snippets and templates

Use the following resources to accelerate development and ensure best practices:

- **Predefined snippets:** Access a library of common SQL patterns and transformations
- **Custom templates:** Create and share organization-specific query templates
- **Version control integration:** Manage and collaborate on SQL code with built-in version control features

By leveraging these advanced querying capabilities, Databricks provides a unified platform that empowers organizations to extract maximum value from their data warehousing investments.



Easily build dashboards with Databricks AI/BI

AI/BI is an innovative tool that combines traditional BI with AI to create a seamless, user-friendly experience for data analysis and dashboard creation. AI/BI features include:

Dual approach	• The dashboards feature a low-code interface for quickly building interactive visualizations and transforming raw data into meaningful insights without extensive coding • The Genie interface provides an AI-powered conversational tool for self-service analytics, enabling business users to ask questions and receive insights instantly
Natural language query support	• Charts can be created using natural language inputs, such as "*Show pipeline by region,*" generating relevant visualizations without complex SQL queries
Flexible configuration options	• A point-and-click interface allows for easy chart customization, while direct SQL editing offers more control for tailored analysis
AI-assisted query explanation	• An inline assistant explains complex queries, detailing data logic and filtering criteria to aid understanding and refinement
Seamless publishing and sharing	• Dashboards can be published easily, with draft and final versions ensuring clarity and accuracy and shared with the team or embedded in external platforms
Advanced data exploration	• Data lineage visualization ensures trust and transparency, while cross-filtering capabilities allow for interactive exploration, uncovering deeper insights
Genie: AI-powered data conversations	• Genie enables ad hoc data queries and dynamic visualization options such as switching between bar and pie charts for flexible data exploration
Semantic understanding and learning	• Business-specific concepts like "churn" can be taught to Genie, which are then consistently applied across queries, enhancing organizational knowledge sharing
Handling complex queries	• AI/BI can answer sophisticated questions, such as *"What was our churn rate last quarter by region?"* and automatically generate visualizations for complex metrics
Data Integrity and accuracy	• AI/BI maintains consistent business logic across queries, ensuring data integrity and providing clear feedback when questions can't be answered

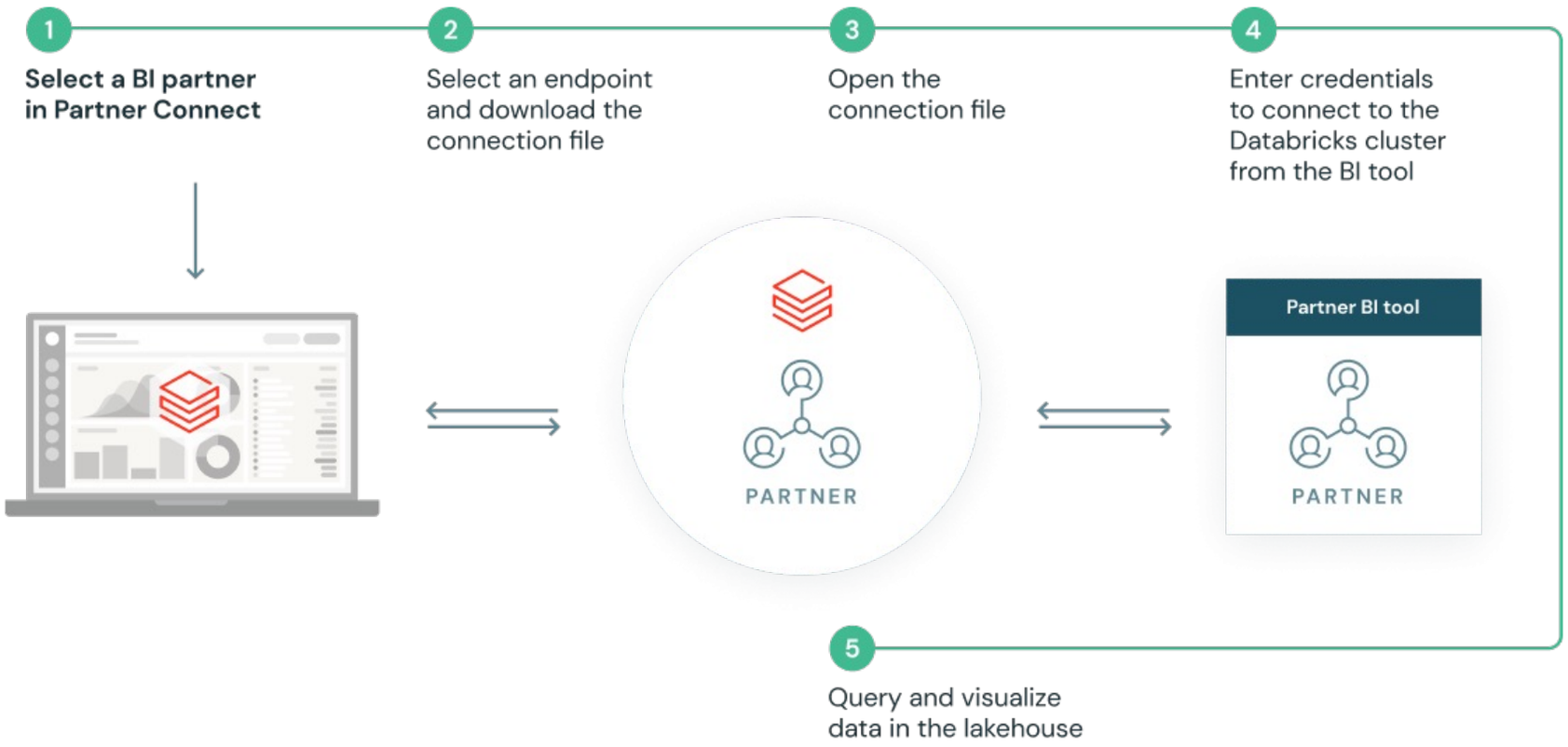
Integrating Databricks AI/BI into the workflow democratizes data access, empowering both technical and nontechnical users to explore data efficiently.

Integrating with external platforms

BI platforms

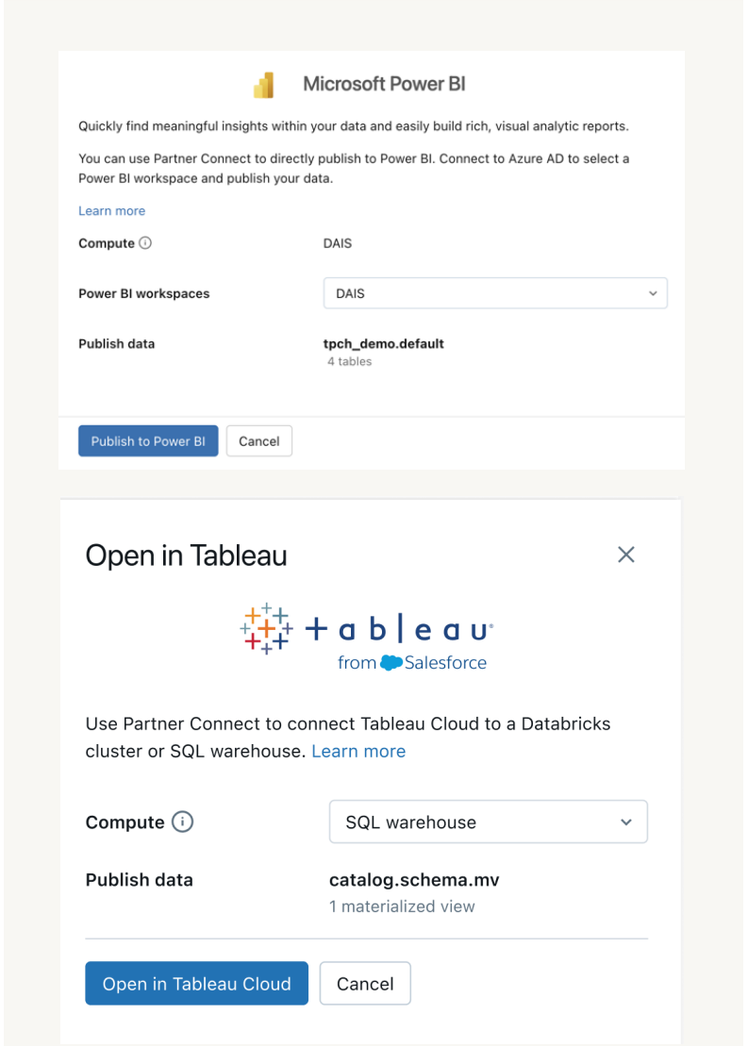
Every business intelligence platform today will soon (if not already) embrace AI-driven data intelligence. It's very likely that your enterprise already has preferred BI platforms such as PowerBI, Tableau, Qlik etc. With an open lakehouse architecture, Databricks SQL can integrate the data in the lakehouse with chosen enterprise BI platforms.

Get started via Partner Connect, which simplifies integration with Databricks SQL by automatically configuring resources — including clusters, tokens and connection files — to connect with partner solutions for business intelligence. The following flow shows how to set up the integration between the BI platform and Databricks SQL in four simple steps:



Databricks works with partner platforms to build deep integrations that allow you to harness the best of a BI platform and the Databricks SQL warehouse. BI platforms integrate with Unity Catalog to get a view of the data assets in the lakehouse, including constraints such as primary keys and foreign keys which allow presenting the relationship between assets seamlessly.

The integration also includes platform-specific integrations. For example, Power BI users can leverage direct query and import modes for different workload profiles, or publish their datasets directly to Power BI for further analysis. Similarly they can leverage live query or extract mode in Tableau and publish their datasets and materialized views directly to Tableau for analysis. The following screenshots from Power BI and Tableau showcase this seamless integration:



It's important to note that all access control measures implemented in Unity Catalog are respected and enforced when accessing data from the BI platforms. This includes advanced fine-grained access control measures such as row-level filtering and column-level masking.

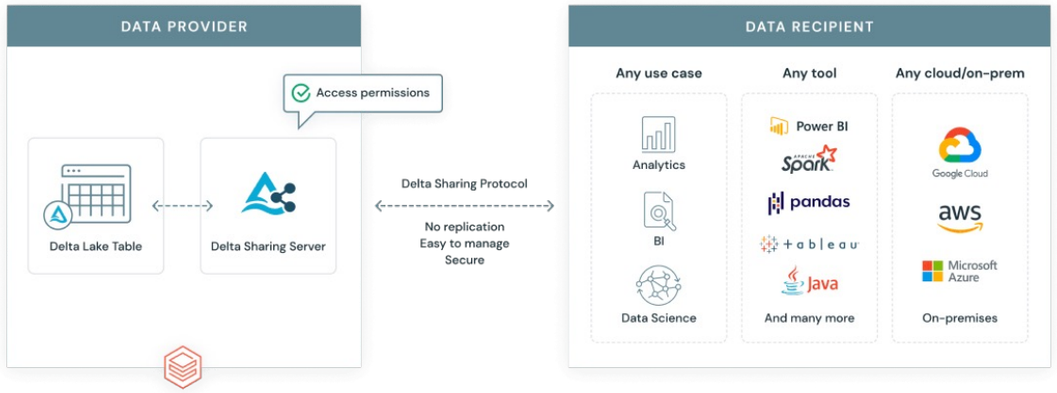
Alternatively, most of the BI platforms also have native Databricks data source connectors to manually configure the integration.

Use the Partner Connect portal to seamlessly integrate preferred BI platforms with serverless Databricks SQL warehouses to get started, or navigate to the assigned serverless Databricks SQL warehouse's "Connection Details" tab to download the respective connection file. The latest integration details for all partners are updated in the Partner Portal.

Data applications

The Databricks open lakehouse architecture provides the power to build data applications which can access the data assets seamlessly with serverless on-demand compute provided by Databricks SQL. These data applications will respect the access control and privacy policies enforced via Unity Catalog. No need to replicate the data to external databases for powering applications.

Databricks SQL warehouses provide JDBC, ODBC, NodeJS, Python, Shell, REST API and many more bindings that can be used to retrieve data from the lakehouse to present in the data applications. The latest list of all bindings and connectors can be found in the SQL drivers and tools portal.



Data sharing and monetization

A key requirement of any warehousing solution is the ability to share data with external organizations for joint analysis or monetization. Databricks SQL uses the open source Delta Sharing protocol as the foundation for key collaboration capabilities: peer-to-peer open cross-platform sharing, zero-ETL and zero-replication sharing, marketplace and data clean rooms.

Peer-to-peer sharing
For peer-to-peer sharing, such as sharing with other business units, recipients using desktop tools or recipients not using Databricks, use Delta Sharing in Unity Catalog. Unity Catalog provides a managed Delta Sharing server that manages authentication of recipients and authorization to access shared data assets.

Data clean rooms
For private, safe collaboration, use data clean rooms to create a secure, isolated, serverless environment for joint data analysis. Clean rooms are built on Delta Sharing, so collaborators don't need to replicate data into the clean room. Clean rooms also provide privacy protection mechanisms such as preventing direct access to data and explicit code approval for execution of analysis.

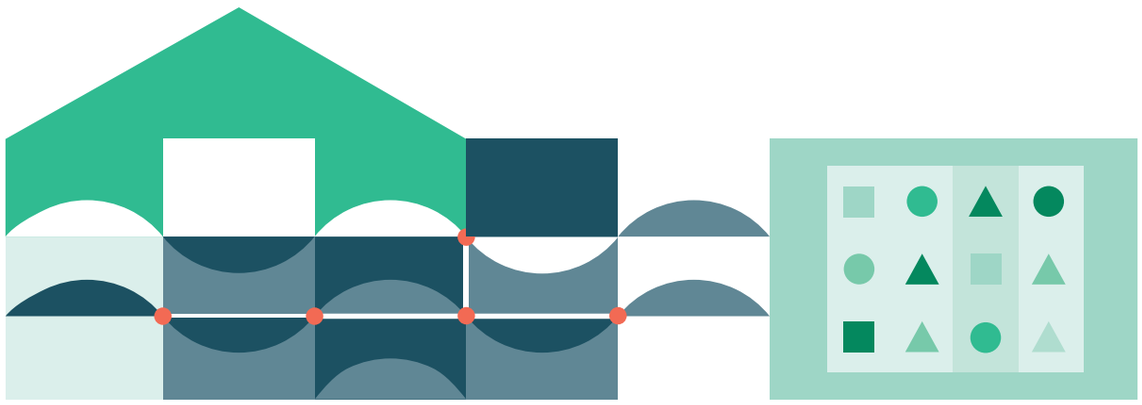
Databricks Marketplace
Databricks Marketplace can be used to publish data assets in a public marketplace or to use data assets published by other providers for public

CHAPTER

04

Migrating to Databricks SQL

An enterprise's data warehouse is where critical data assets are stored, managed and accessed. It plays an important role in consolidating information from various sources across the organization, including transactional systems, operational databases and external data sources.



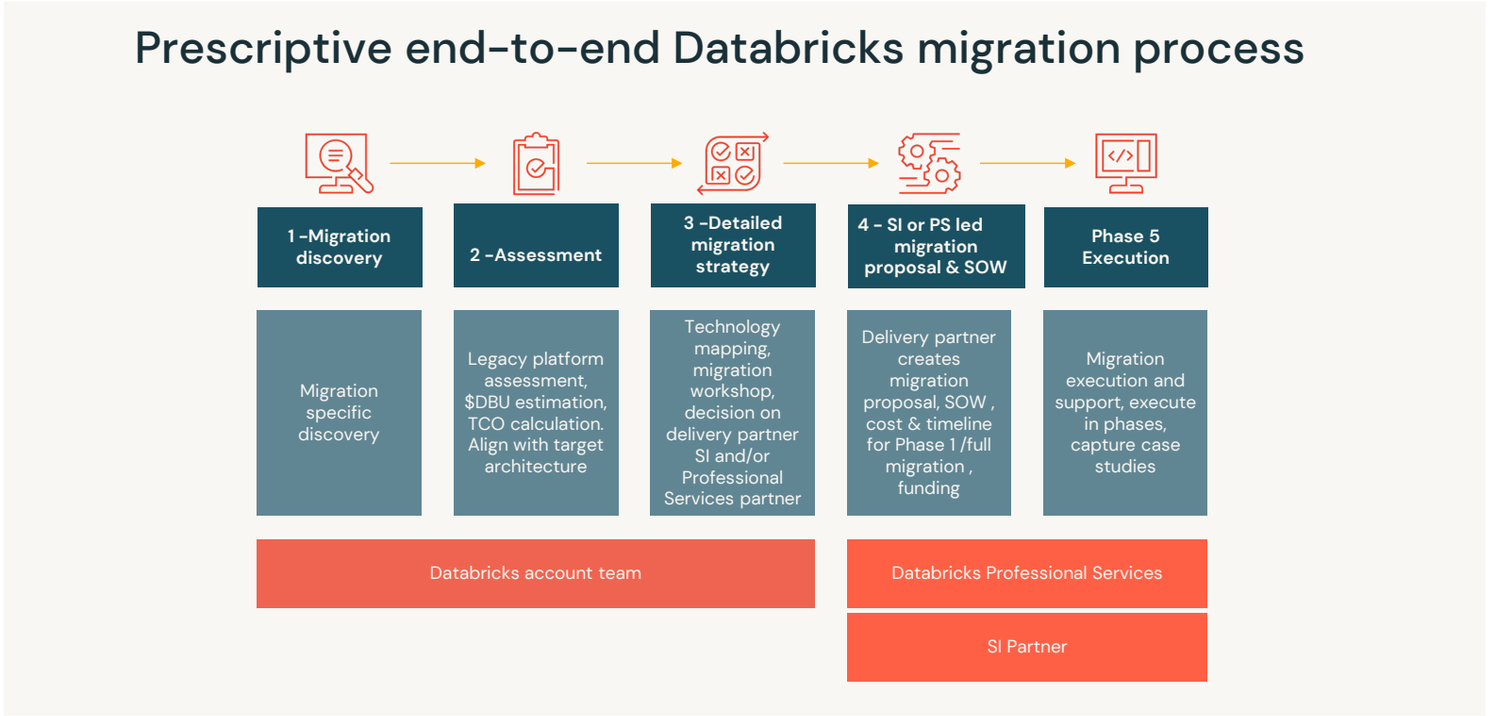
Providing a single source of truth is very important for enterprise data warehouses, empowering decision-makers with accurate, timely and actionable insights — ultimately generating data-driven decisions across the enterprise.

Before Databricks lakehouse, the traditional on-premises or cloud data warehouse solution usually created many data silos, lacked unified governance, took too long to load and process, lacked scalability and didn't support all data workers (e.g., data engineers, data scientists, data analysts, business users) with a single platform.

Migrating an enterprise data warehouse can seem daunting, as data warehouses can store multiple years, if not decades of historical data. With many years of experience helping customers migrate from enterprise data warehouses to the Databricks Platform, Databricks has built a proven migration methodology and best practices to make the migration journey seamless, simple and scalable.

Migration process

Here is the prescriptive migration process recommended by Databricks:



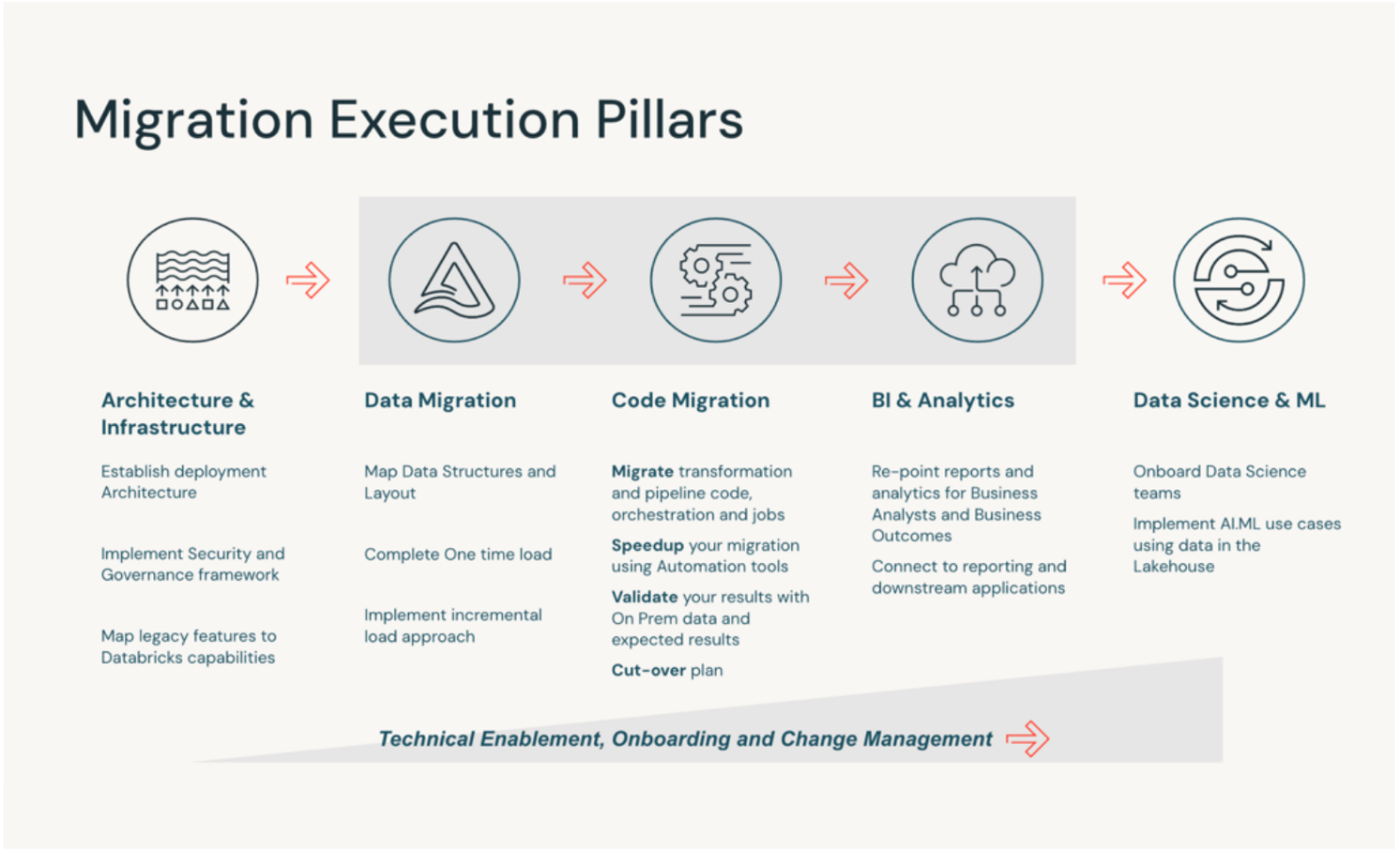
- **Use-case discovery:** This involves identifying the right source systems to migrate or modernize. It's crucial to understand the current architecture and the desired state. A strong business use case will get enough support from the stakeholders, prove the value quicker and pave the way for migration of future use cases.

- **Assessment:** This involves a detailed legacy platform assessment, target architecture, Databricks Unit cost (DBU$) calculation and total cost of ownership (TCO) and business value assessment analysis. There are various tools available for assessment.

- **Migration strategy readout and win:** This stage involves preparing and finalizing the Databricks target architecture, deciding on ingestion patterns, finalizing the ETL pattern and selecting a delivery partner. Selecting the right partners or Databricks professional services are critical for the migration process, as the team will have the automated tools, technical guidance, partner solutions and professional services to help you eliminate risk and realize value faster.

- **Migration proposal and statement of work (SOW):** The chosen delivery partner prepares a detailed migration proposal with cost and timelines. The migration proposal and SOW need to be carefully evaluated by Databricks migration subject matter experts or the professional services team so that any potential challenges and roadblocks are identified and a mitigation plan is in place to ensure project success.

Execution: This is the final stage where the migration execution and support take place. There are various stages during the migration execution. Databricks Professional Services and certified migration partners play a crucial role in providing migration advice and assurance. It's also important to engage in technical enablement during the migration to ensure that best practices are followed, facilitating a smooth and successful migration.



Migration strategy

Here are the common migration options.

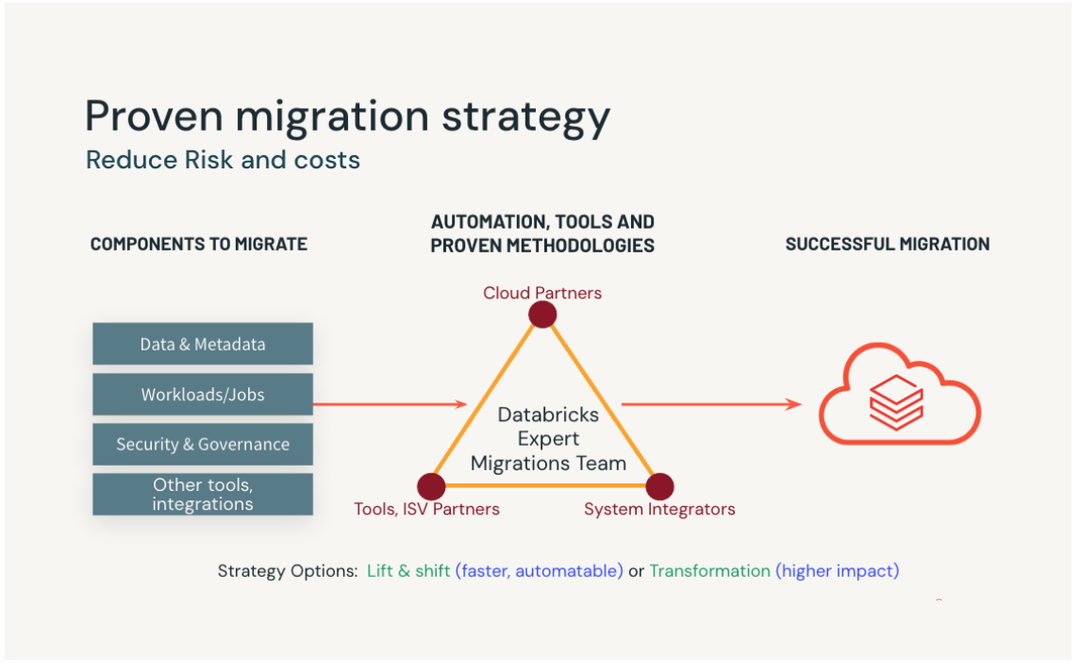
- **Lift and shift**: This approach involves moving data and workloads with minimal changes. It's faster but it could carry the legacy workloads from the current data warehouse and might not fully leverage Databricks capabilities. Most auto-conversion tools support converting the existing code and ETL to Databricks SQL, jobs or PySpark code with certain conversion success rate.

- **Replatforming:** Refactor existing ETL processes and workloads to take advantage of Databricks distributed computing and advanced analytics features. It takes more time compared with the lift and shift approach, however it helps to refactor the existing code and ETL according to Databricks best practices.

- **Incremental migration:** Migrate in phases, starting with less critical workloads or datasets to minimize risk and impact on operations.

- **Net new use cases:** Create new use cases and patterns based on best practices from Databricks lakehouse architecture, and then migrate the existing use cases based on the new patterns.

Each option has pros and cons. Please work closely with the Databricks migration experts or certified partners to ensure a successful migration and reduce risk and costs during the entire migration.



Migration tools

There are a number of required tools to help accelerate migration. At a high level, the tools usually cover the following key capabilities:

1. **Assessment:** Collect and analyze the code complexity and assess the migration efforts
2. **Conversion:** Convert the source code automatically to Databricks SQL, PySpark or workflows
3. **Reconciliation:** Reconcile the data

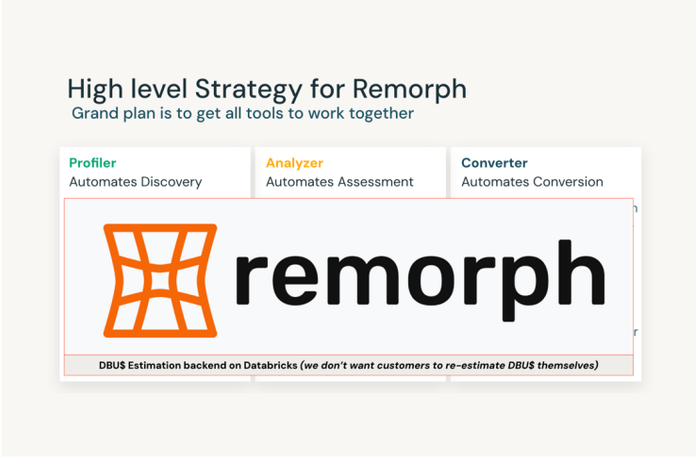
The current choice of tools are BladeBridge, Remorph and Lakehouse Utils. Here is a brief introduction for each:

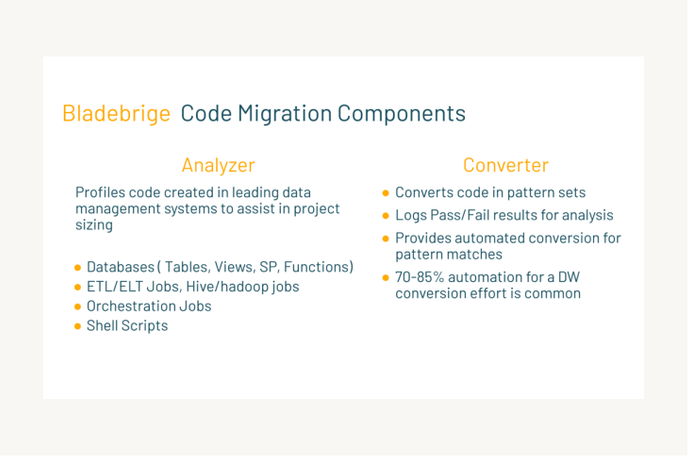
Remorph
Remorph is a comprehensive toolkit meticulously crafted to facilitate seamless migrations to Databricks. It is available for all customers and Databricks regularly releases small, incremental functional releases approximately every month. Remorph aims to provide a self-service code migration experience for customers who want to move away from other SQL dialects. This command-line tool simplifies and optimizes the entire migration process. Currently, it covers two major user journeys related to converting SQL code (transpile) and verifying data migration correctness (reconcile).

- **Transpile:** Converts other SQL code into working Databricks SQL code. It currently supports Snowflake and is actively working on adding Transact SQL (Synapse and SQL Server).
- **Reconcile:** Identifies discrepancies and variations in data when comparing the source with the Databricks target. It currently supports Snowflake and Oracle platforms.

BladeBridge
BladeBridge is a tool that facilitates the migration of data warehouses to modern platforms like Databricks. It automates the process of converting legacy SQL code, ETL processes and data structures into formats compatible with Databricks, significantly reducing the time and effort required for migration.

BladeBridge supports the translation of different SQL dialects and ETL workflows, ensuring that existing data pipelines and queries can be seamlessly adapted to the Databricks environment. This automation helps to minimize errors, streamline the migration process and ensure a smoother transition to Databricks.





Lakehouse Utils
dbt is a popular tool used by many customers. Databricks has Lakehouse Utils to migrate dbt (Snowflake or Redshift) to dbt in Databricks. The purpose of the Lakehouse Utils package is threefold:

- **Streamline migration:** Reduces the time and effort needed to migrate pipelines from cloud data warehouses to the lakehouse (e.g., dbt and Databricks) by converting non-native functions to compatible Apache Spark™ SQL functions using dbt macros or regex conversions.
- **Centralized function mapping:** Serves as a single source for mapping warehouse functions to Databricks functions, highlighting cases where manual intervention is needed. A full list of supported functions is provided in the functionlist.csv.
- **Ensure reliability:** It promotes best practices in unit testing to ensure the robustness and reliability of the macros.

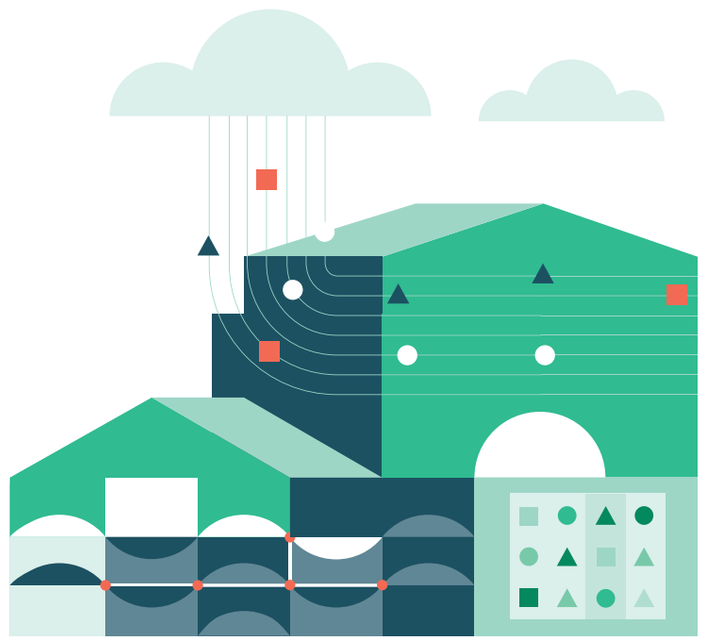
Lakehouse Utils provides a simplified user experience for migration, so users can convert SQL code directly where their existing project lives. To run a code migration, users just need to follow these steps:

- Add lakehouse_utils as a package to your dbt project
- Run "dbt deps"
- Identify your source model and data warehouse and run the convert_to_databricks script from the command line

The Databricks Platform modernizes data management, overcoming limitations from the traditional data warehouses. With proven migration strategies and tools like Remorph and Lakehouse Utils, transitioning to Databricks becomes seamless and scalable. This empowers organizations to enhance analytics capabilities, enabling data-driven decision-making across the enterprise.

Next steps

1. Please reach out to your Databricks account team. They'll connect you to migration experts who can provide best practices and recommendations for your specific migration use case.
2. Please refer to Migrate to Databricks for migration steps, case studies and Brickbuilder Solutions to accelerate your migration journey.
3. Check the Data Warehouse Brickbuilder Migration Solutions to see if a partner solution can help you migrate your data warehouse.

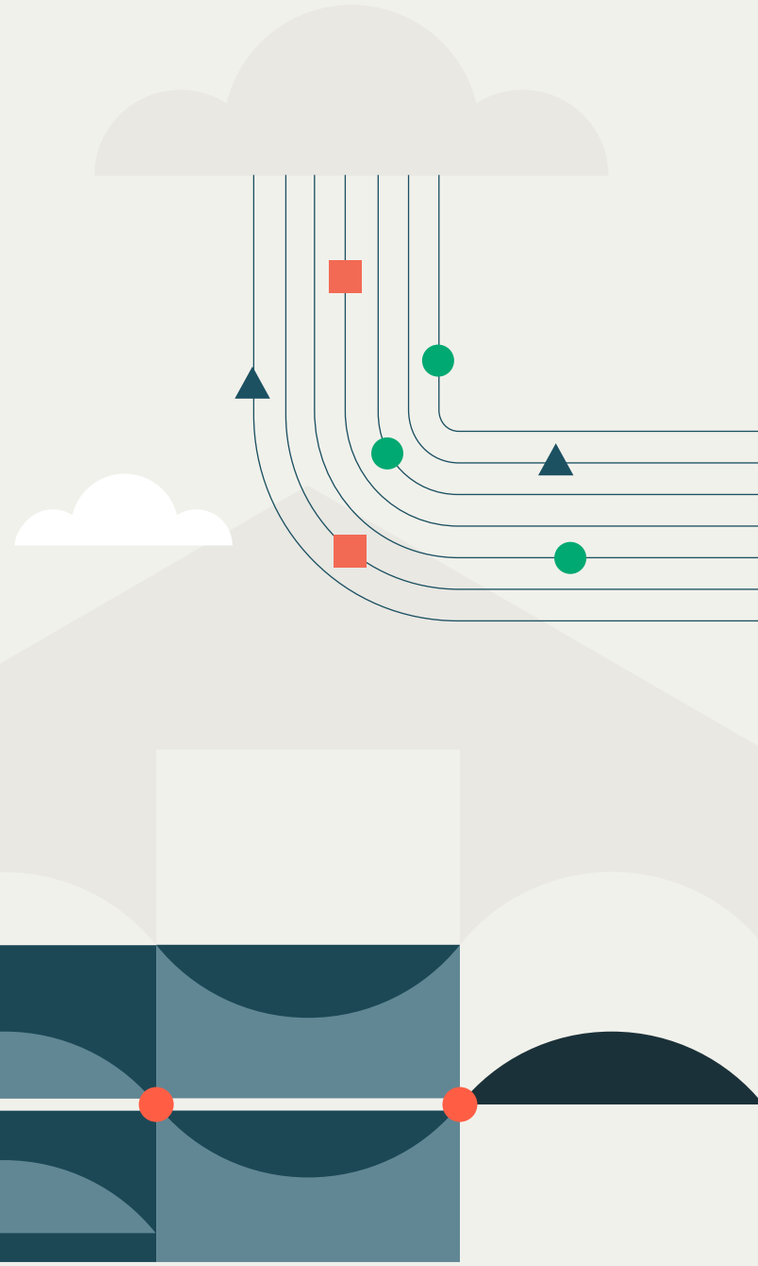


About Databricks

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



databricks