databricks

Feature Store

LECTURE

Introduction to Feature Store

In this lecture, we introduce the feature store, covering what it is, the Databricks Feature Store, its unified permission and data lineage models, and the differences between Workspace and Unity Catalog feature stores.

What is a Feature Store?

A **feature store** manages features, or input data to a machine learning model.

In a model that predicts **customer churn**, for example, features could be:

- Aggregations of raw data over time windows, like **trailing 7-day purchases**
- Joined **combinations of data sets**, like customer demographic information joined to transaction features
- Complex functions of customer information, like **estimated customer lifetime value**

The process of creating these values from data is **feature engineering**.

A feature store is a system that manages features—these are the input variables used in machine learning models.
It helps organize, store, and reuse features so that they can be consistently used across different models.
For example, in a customer churn model, features might include:

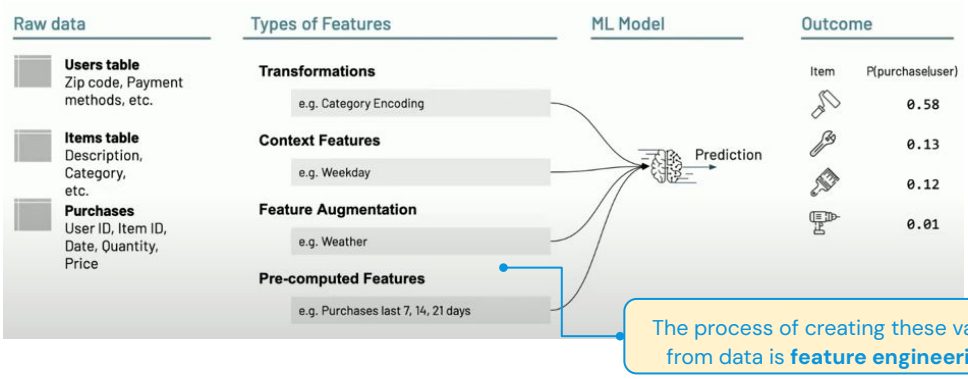
- Aggregated values like trailing 7-day purchases
- Joined data from multiple sources, such as demographics and transactions
- Calculated fields like estimated customer lifetime value

Creating these types of values from raw data is part of feature engineering.

What is a Feature Store?

An example of a recommendation system.

A **feature store** manages features, or input data to a machine learning model.



This example shows how a feature store works in the context of a recommendation system.
Raw data comes from different tables like users, items, and purchases.
These raw inputs are transformed into features, such as:

- Encoded categories
- Weekday information
- External data like weather
- Pre-computed metrics, like purchases in the last 7, 14, or 21 days

These features are used by the ML model to make predictions, like the probability of a user purchasing an item.
All these features are managed and served through the feature store, and creating them is part of feature engineering.

Why Would You Need a Feature Store?

Basic Motivations

Discovery 

Multiple Data Scientists are trying to solve similar modeling tasks and come up with different definitions of the same features. **How can I find the features**?

Lineage 

Model governance requires documentation of the features used to train a model, as well as the **upstream lineage** of a feature to reliably use it. **How is it computed, and who owns it**?

Skew 

When multiple teams manage feature computation and ML models in production, minor yet significant **skew in upstream data** at the input of a feature pipeline can be very hard to detect and fix.

Online Serving 

During exploration and model experimentation phases features are implemented in frameworks that do not scale to production.

Why use Databricks Feature Store?

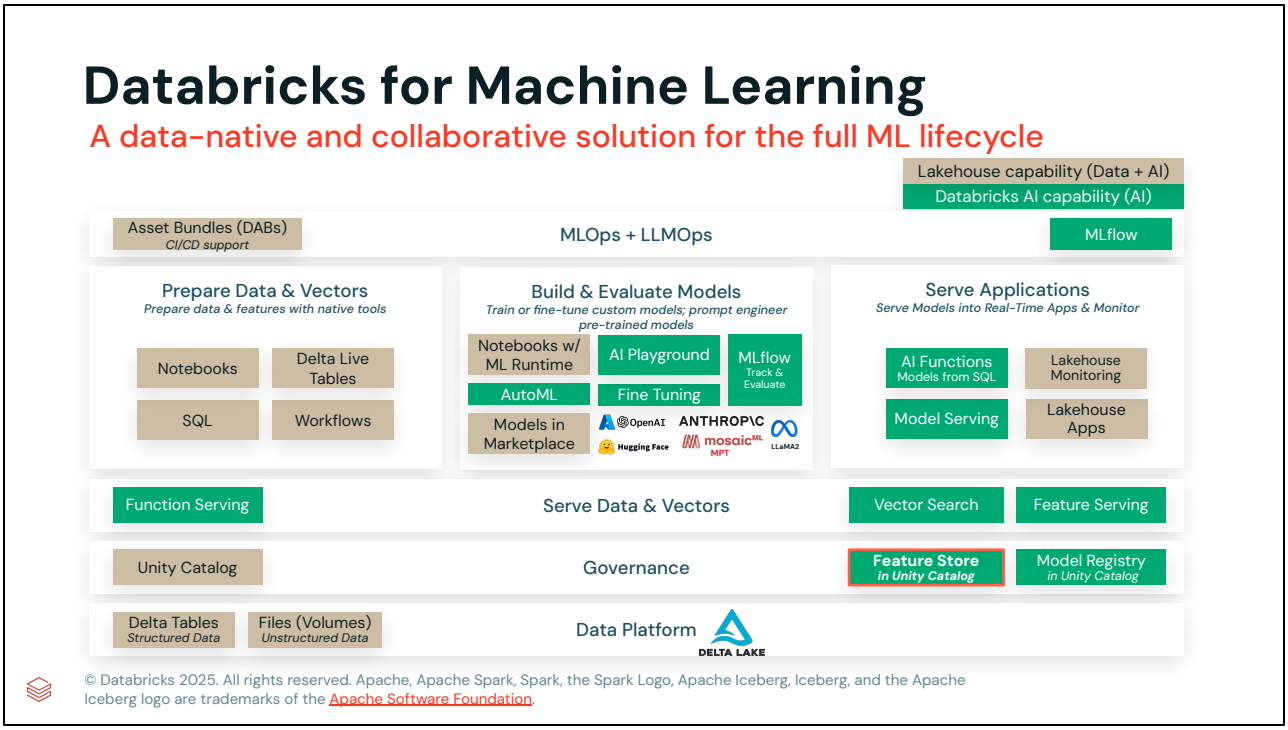
Databricks Feature Store is fully integrated with other components of Databricks.

Discoverability. The Feature Store UI, accessible from the Databricks workspace, lets you browse and search for existing features.

Lineage. When you create a feature table with Feature Store, the data sources used to create the feature table are saved and accessible. For each feature in a feature table, you can also access the models, notebooks, jobs, and endpoints that use the feature.

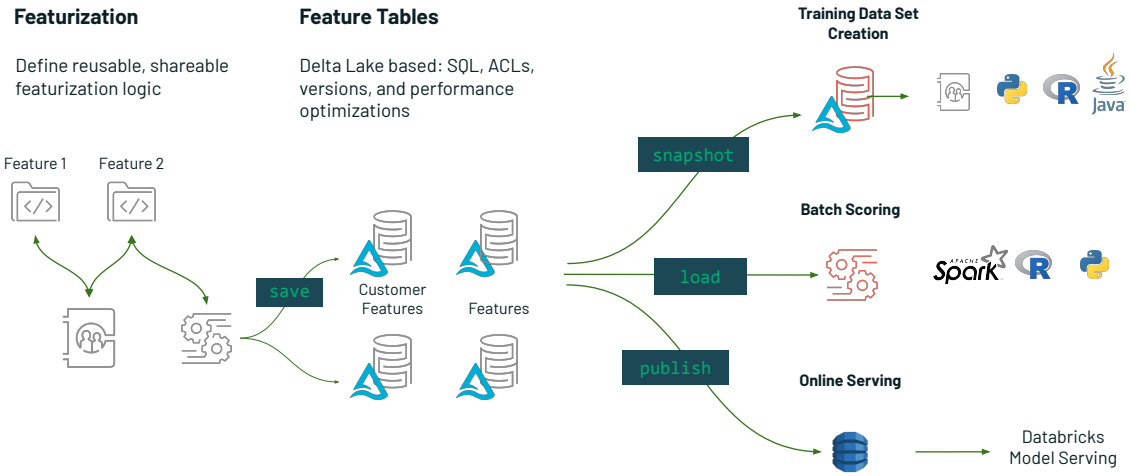
Skew. When multiple teams manage feature computation and ML models in production, minor yet significant skew in upstream data at the input of a feature pipeline can be very hard to detect and fix.

Online serving. During exploration and model experimentation phases features are implemented in frameworks that do not scale to production.



In the Databricks stack, the feature store is now integrated as part of Unity Catalog. A feature engineering client, provided as a Python API, allows you to incorporate DataOps practices into your feature engineering pipeline and interact with or create feature tables.

Databricks Feature Store



How does Databricks Feature Store work?

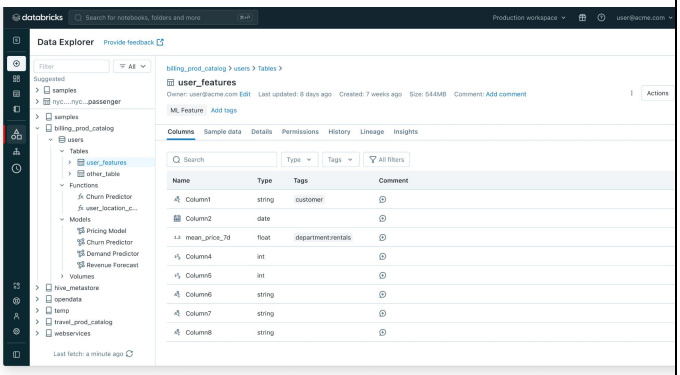
Write code to convert raw data into features and create a Spark DataFrame containing the desired features. Then, save features to feature tables that are based on Delta Lake supporting ACLs, versioning and performance optimization.

Then, these feature tables can used for model training, batch inference and real-time inference.

Complete Integration-FS with Unity Catalog

Any table can be a feature table

- Feature Tables become regular UC Tables with additional metadata.
- Shared properties are unified.
  - Feature table description == table comment.
  - Feature table schema == table schema
- Three-level namespace convention



Feature Store integrates fully with Unity Catalog, meaning any Unity Catalog table can be used as a feature table.
Feature tables behave like regular Unity Catalog tables but include additional metadata for feature management.
The shared properties are unified:

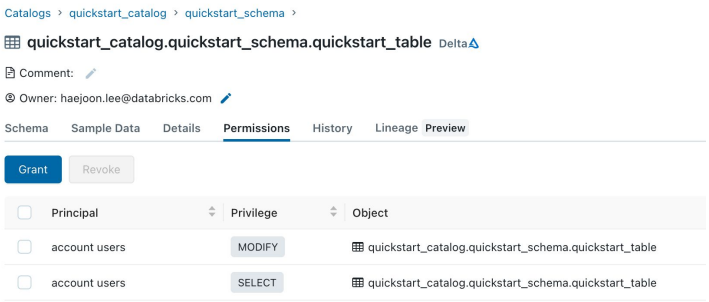
- The feature table description matches the table comment.
- The feature table schema matches the table schema.

It also follows the three-level namespace convention—catalog, schema, and table—which ensures consistency across the platform.

Unified Permission Model

Secure features with built-in governance

- Feature data and metadata are governed by the Unity Catalog permission model.
- Future improvements to data governance in Unity Catalog will apply to Feature data.



Feature Store uses Unity Catalog's permission model to manage access to both feature data and metadata.

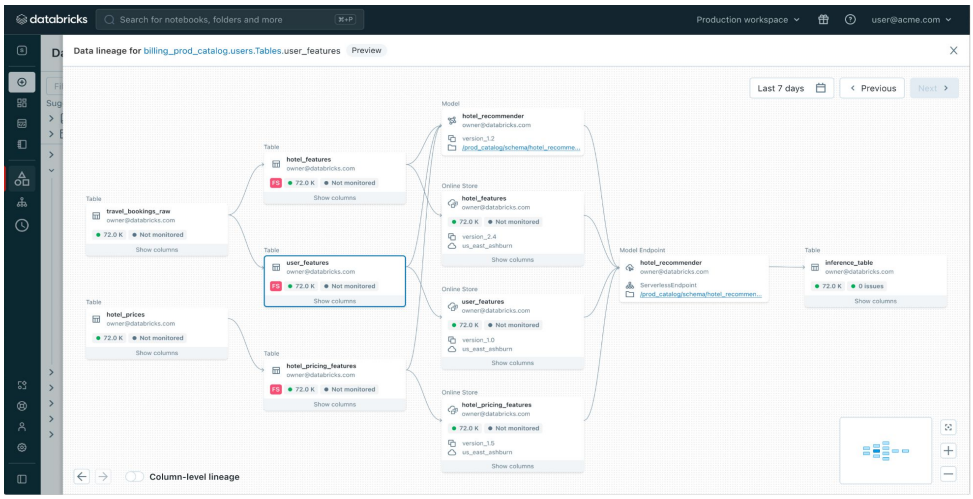
This ensures that the same security and governance rules used for other Unity Catalog tables also apply to feature tables.

Any future updates to Unity Catalog's governance capabilities will automatically extend to feature data as well.

An example of permission settings, where specific privileges like MODIFY or SELECT are granted to different user groups.

Unified Data Lineage

Feature Store Lineage and Unity Catalog lineage become one



Feature Store lineage is fully integrated with Unity Catalog lineage.
This means feature tables, their source data, and downstream usage can all be tracked in one place.
The lineage view helps trace how data flows—from raw inputs through transformations to the final feature tables and models.
This supports better understanding of dependencies and ensures consistency across the workflow.

Integration with MLflow and Model Serving

Makes model deployment easier

Integration with MLflow

When a features from FS are used for training, **the model is packaged with feature metadata**.

In inference time, the model can look up features at runtime.

Integration with Model Serving

When the model is used for batch scoring or online inference, **it automatically retrieves features from Feature Store**.

Feature Store integrates with both MLflow and Model Serving to support model deployment.
With MLflow, when features from Feature Store are used during training, the model is packaged along with the feature metadata.
At inference time, the model can look up the required features using that metadata.
With Model Serving, when a model is deployed for batch or online inference, it automatically retrieves the needed features from the Feature Store.

Workspace FS vs. Unity Catalog FS

Workspace FS

- Specific to a single workspace
- Challenges in sharing feature tables across workspaces
- Uses `FeatureStoreClient` to create feature tables.

Unity Catalog FS (Recommended)

- Any Delta table with a primary key can be a feature table (DBR 13.2+)
- All UC capabilities are available; discovery, governance, lineage and cross-workspace access.
- Uses `FeatureEngineeringClient` to create feature tables.

Workspace Feature Store is limited to a single workspace and can be harder to use when sharing feature tables across workspaces. It uses the FeatureStoreClient to create feature tables.

Unity Catalog Feature Store allows any Delta table with a primary key to be used as a feature table, starting from DBR 13.2.
It supports Unity Catalog features like discovery, governance, lineage, and access across workspaces.
It uses the **FeatureEngineeringClient** to create feature tables.

Thank you for completing this lesson and continuing your journey to develop your skills with us.