O'REILLY®

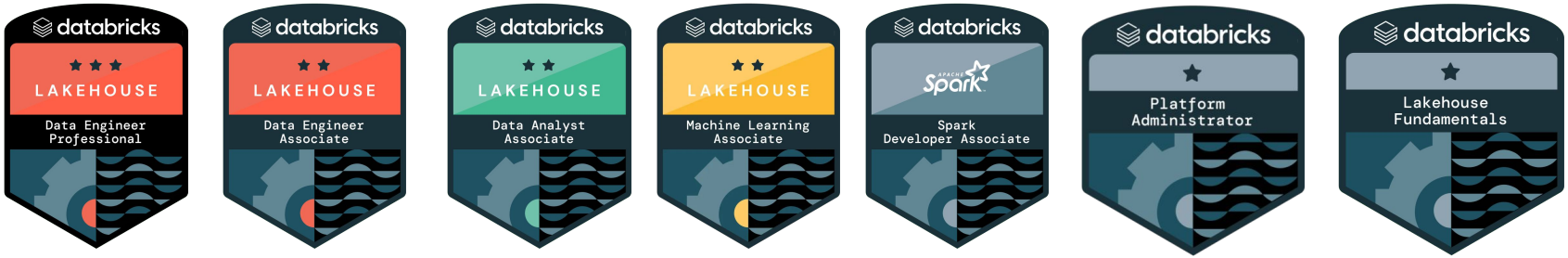
Databricks Data Engineer Associate Certification Prep in 2 Weeks

Derar Alhussein

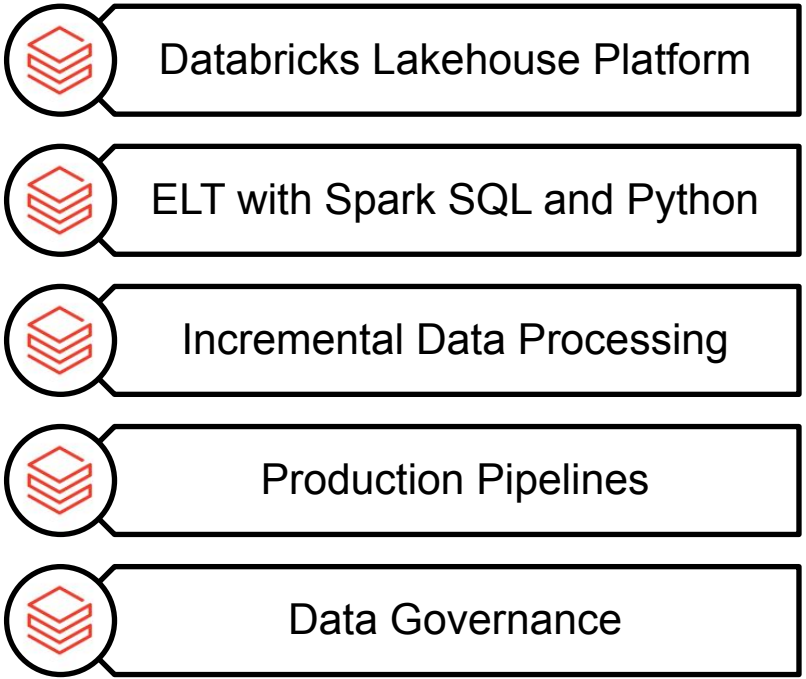
Derar Alhussein

- Senior Data Engineer
- Master’s degree in Data Mining
- Udemy Instructor (25K students)

www.linkedin.com/in/DerarAlhussein



Course outline



Schedule

- **Day 1: Databricks Lakehouse Platform**
  - Databricks Workspace, Notebooks, Repos, Delta Lake
- **Day 2: ETL with Spark SQL and Python**
  - Relational entities, Querying data files, Advanced transformations
- **Day 3: Incremental Data Processing**
  - Spark Structured Streaming, Multi-hop architecture, DLT, CDC
- **Day 4: Production Pipelines and Data Governance**
  - Databricks Jobs, DBSQL, Data objects privileges, Unity Catalog

Prerequisites

- SQL language
- basic Python

Poll

- How familiar are you with Databricks ?

A. Very familiar

B. Somewhat familiar

C. Not at all familiar

Day 1 Agenda

- Introducing Databricks
- Setting up Databricks workspace
- Exploring Databricks Workspace
- Clusters
- Working with Notebooks
- Databricks Repos

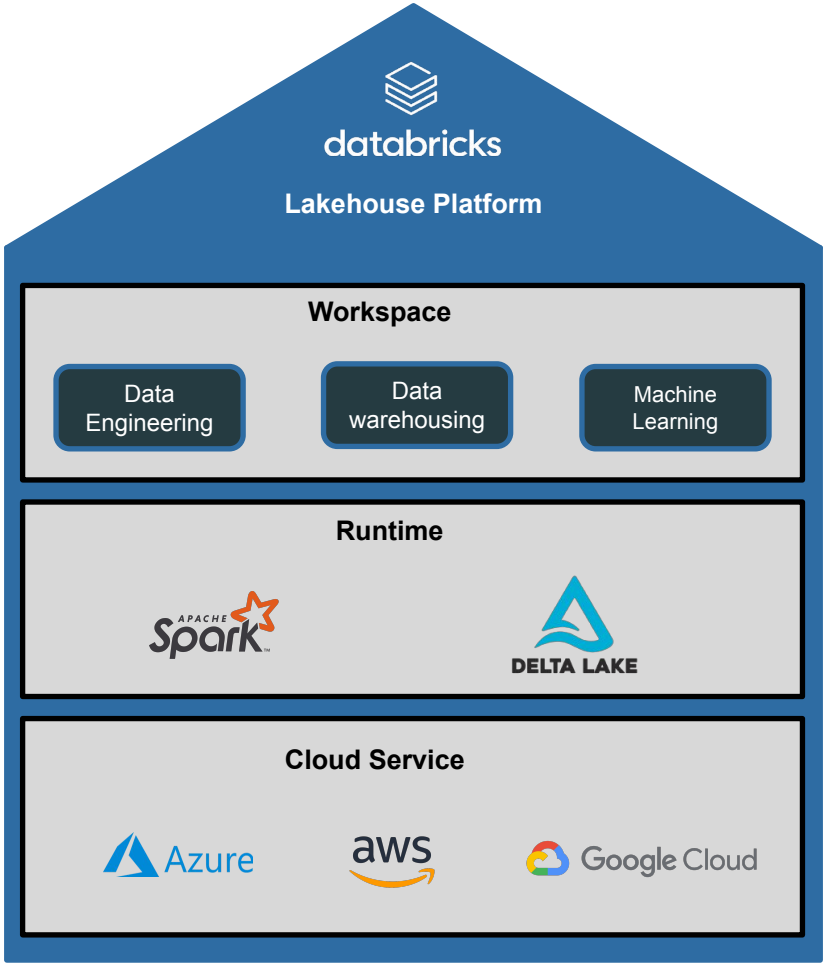
What is Databricks

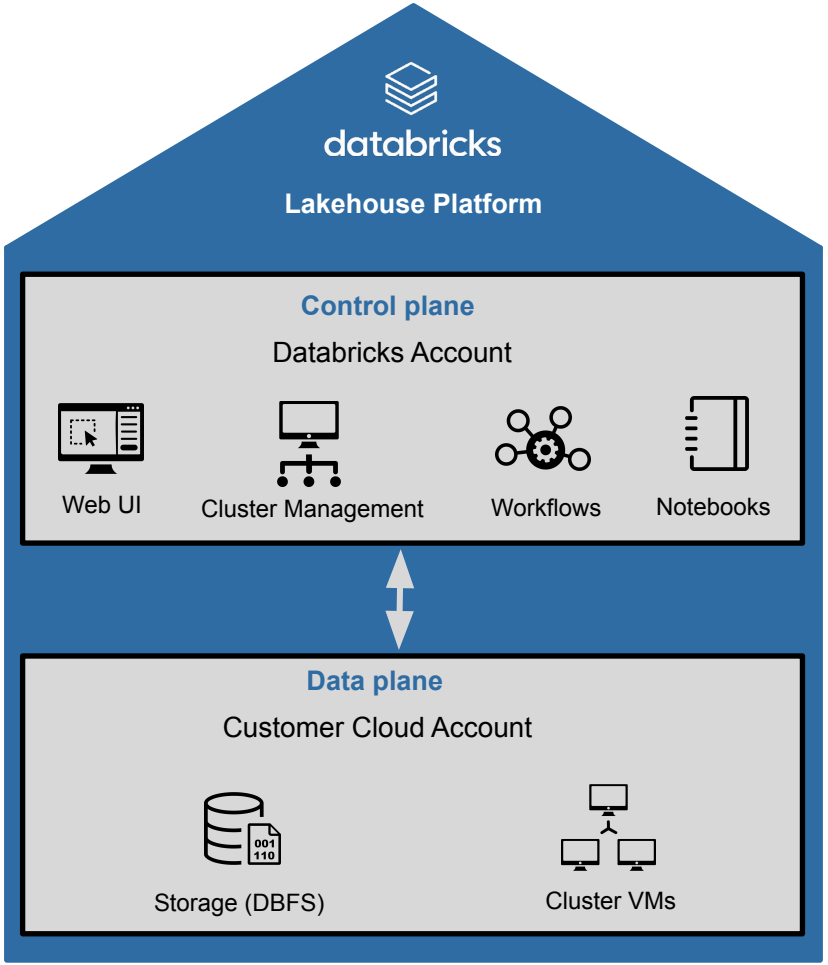
Databricks

- Multi-cloud Lakehouse Platform based on Apache Spark

Lakehouse


Data Lake
- Open
- Flexible
- ML support
Lakehouse
All your data engineering, Analytics, and AI workloads are in one platform
Data warehouse
- Reliable
- Strong governance
- Performance


databricks
Lakehouse Platform
Workspace
Data Engineering
Data warehousing
Machine Learning
Runtime
APACHE Spark
DELTA LAKE
Cloud Service
Azure
aws
Google Cloud


databricks
Lakehouse Platform
Control plane
Databricks Account
Web UI
Cluster Management
Workflows
Notebooks
Data plane
Customer Cloud Account
001
110
Storage (DBFS)
Cluster VMs

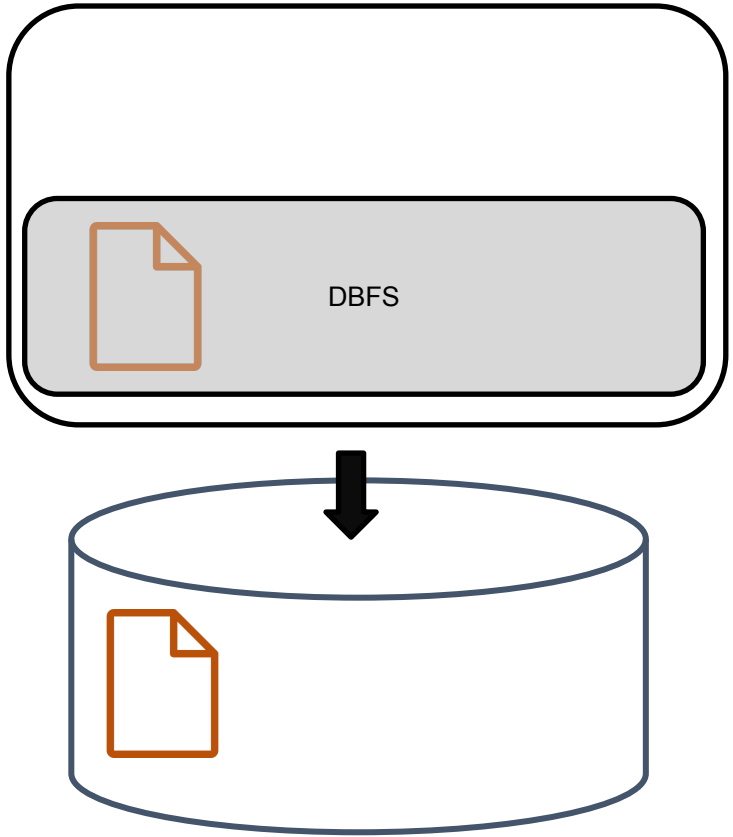
Spark on Databricks

- In-memory, distributed data processing
- All languages support
    - Scala, Python, SQL, R, & Java
- Batch processing & stream processing
- Structured, semi structured, & unstructured data

Databricks File System (DBFS)

- Distributed file system
- Preinstalled in Databricks clusters
- Abstraction layer
  - data persisted to the underlaying cloud storage

Cluster



Cloud Storage

Poll

- According to the Databricks Lakehouse architecture, which of the following locations completely hosts the customer data ?

A. Control plane

B. Databricks account

C. Databricks Runtime

D. Customer's cloud account

E. Workspace

Poll

- According to the Databricks Lakehouse architecture, which of the following locations completely hosts the customer data ?

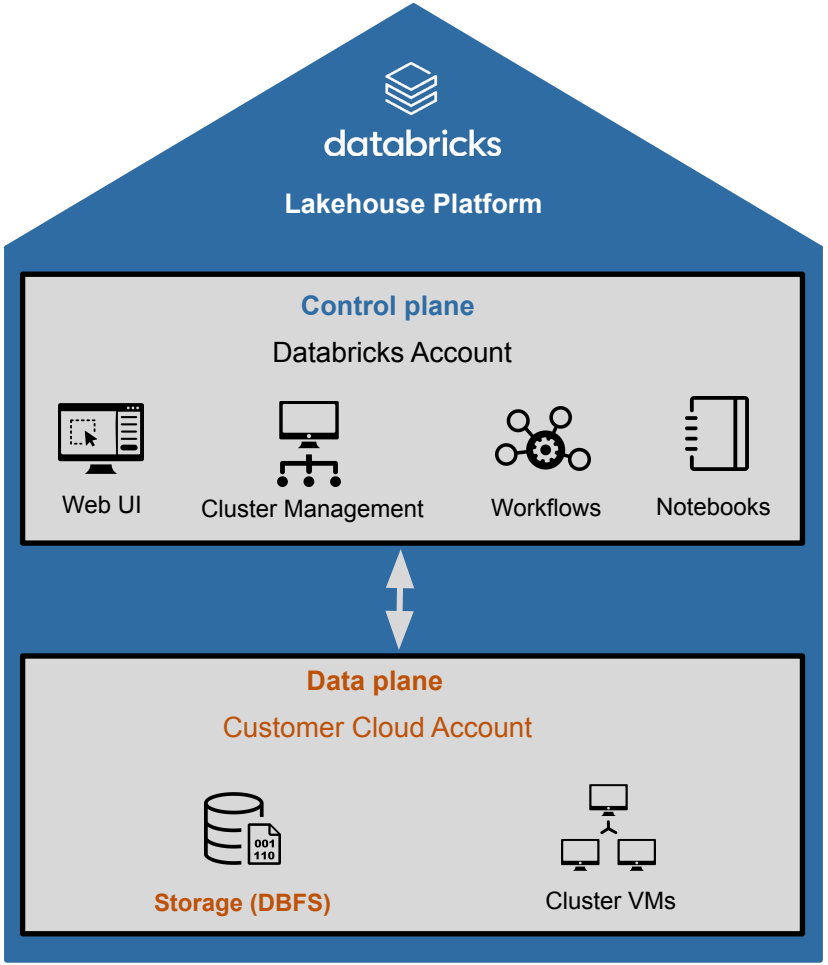
A. Control plane

B. Databricks account

C. Databricks Runtime

D. Customer's cloud account (Data Plane)

E. Workspace


databricks
Lakehouse Platform
Control plane
Databricks Account
Web UI
Cluster Management
Workflows
Notebooks
Data plane
Customer Cloud Account
001
110
Storage (DBFS)
Cluster VMs

Q&A

Setting up Databricks workspace

Poll

- What is your cloud provider ?

A. Azure

B. AWS

C. GCP

D. Other / I don’t have a cloud account

Presentation: Set up Databricks workspace

- Azure: https://portal.azure.com
- AWS
  - Option 1: Databricks website https://www.databricks.com/try-databricks
  - Option 2: AWS marketplace https://aws.amazon.com/marketplace/pp/prodview-wtyi5lgtce6n6
- GCP: https://console.cloud.google.com/marketplace/product/databricks-prod/databricks

Presentation: Community Edition

- Sign up: https://www.databricks.com/try-databricks
- Login: https://community.cloud.databricks.com/

Pricing

- https://www.databricks.com/product/pricing

Hands-on Lab

- Create your Databricks workspace in your own cloud account

Estimated Time: 15 Minutes

- Create your Databricks workspace in your own cloud account

Q&A

Exploring Databricks Workspace

Presentation

- Navigating workspace
- Import course materials

Hands-on Lab

- Explore your Databricks Workspace
- Import course materials from Github into your workspace

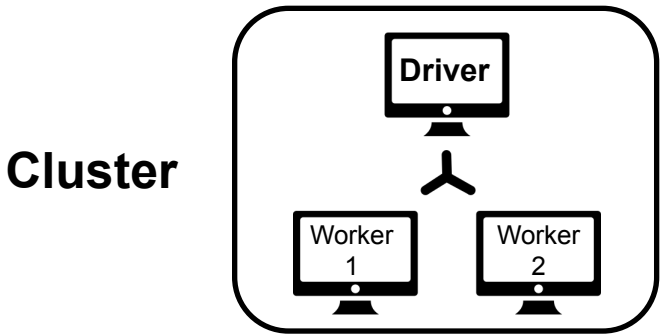
Estimated Time: 10 Minutes

- Create your Databricks workspace in your own cloud account

Clusters

Cluster Definition

- Set of computers (AKA nodes or instances) working together like a single entity.
- It consists of a master node (the driver) and some other worker nodes.
  - The driver node is responsible for coordinating the workers



Tables

All-purpose cluster

- Used to run interactive notebooks for development and analyzing data.
- Can be created using the Web UI, Command Line (CLI), or REST API
- Can be manually terminated, or auto terminated after X minutes of inactivity

Job cluster

- Used to run automated jobs
- Created automatically by Databricks job scheduler
- Terminate when the job is completed

Databricks pools

- Databricks pools are a set of idle, and ready-to-use instances.
- Help reducing cluster start and auto-scaling times
- Databricks does not charge any fee for idle instances in the pool. While, the cloud provider will charge you for these running instances.

Presentation

- Creating Cluster

Hands-on Lab

In your Databricks workspace:

- Solve Lab: **1.0L - Creating Clusters**

Estimated Time: 10 Minutes

- Create your Databricks workspace in your own cloud account

Q&A

Working with Notebooks

Presentation

- 1.1 - Notebook Basics

Hands-on Lab

In your Databricks workspace:

- Solve Lab: **1.1L - Notebook Basics**

Estimated Time: 15 Minutes

- Create your Databricks workspace in your own cloud account

Q&A

Databricks Repos

Presentation

- GIT integration
- Creating branches, Push and Pull changes

Poll

- Which of the following functionalities can *__Not__* be performed in Databricks Repos ?

A. Clone a remote Git repository

B. Push to, and Pull from a remote Git repository

C. Create pull requests

D. Create and manage branches

E. Display differences upon commit

Poll

- Which of the following functionalities can *__Not__* be performed in Databricks Repos ?

A. Clone a remote Git repository

B. Push to, and Pull from a remote Git repository

C. Create pull requests

D. Create and manage branches

E. Display differences upon commit

Q&A

Tomorrow …

- Delta Lake
- Advanced Delta Lake Features
- Relational Entities on Databricks
- Set Up Delta Tables
- Views
- Querying data files

O'REILLY®

Databricks Data Engineer Associate Certification Prep in 2 Weeks

Day 2

Derar Alhussein

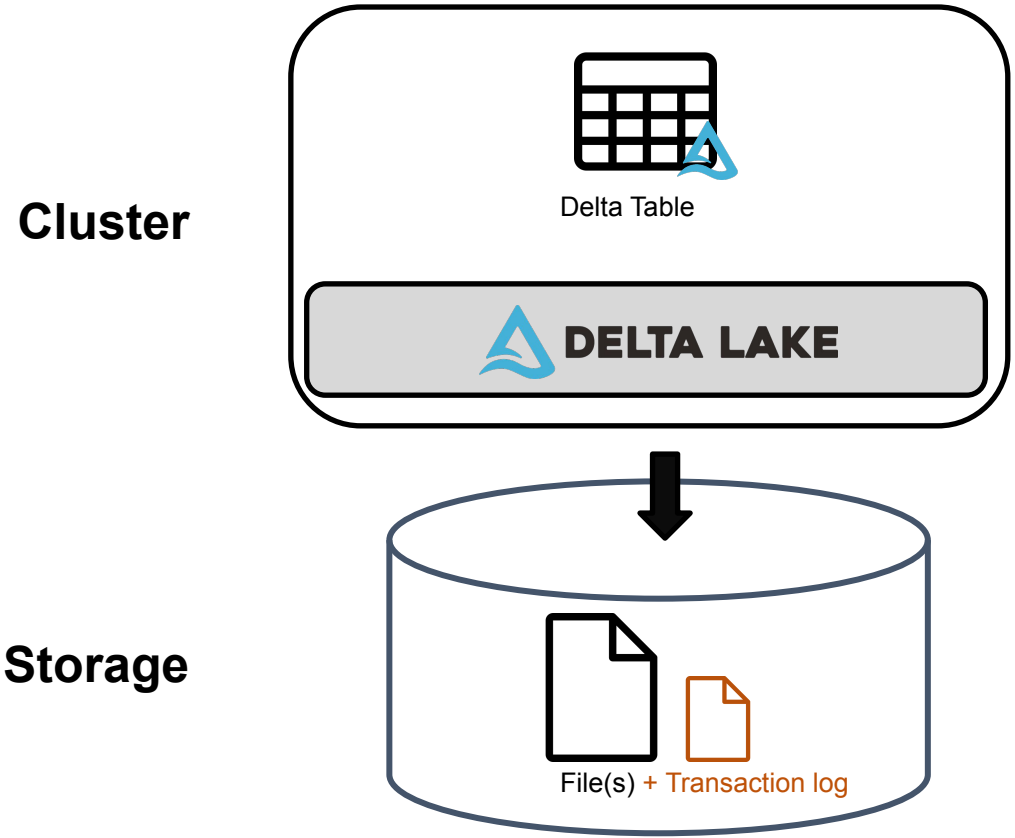
Day 2 Agenda

- Delta Lake
- Advanced Delta Lake Features
- Relational Entities on Databricks
- Set Up Delta Tables
- Views
- Querying data files

Delta Lake

Delta Lake

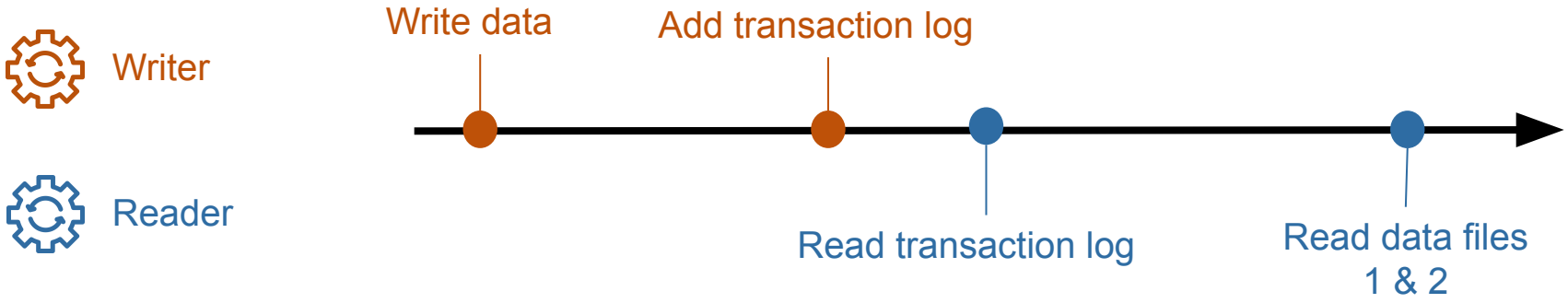
- Open-source storage framework that brings reliability to data lakes


Cluster
Delta Table
DELTA LAKE
Storage
File(s) + Transaction log

Transaction log (Delta log)

- Ordered records of every transaction performed on the table
- Single Source of Truth
- JSON file contains commit information:
  - Operation performed + Predicates used
  - data files affected (added/removed)

Writes/Reads



Data Files

File 1.parquet

File 2.parquet

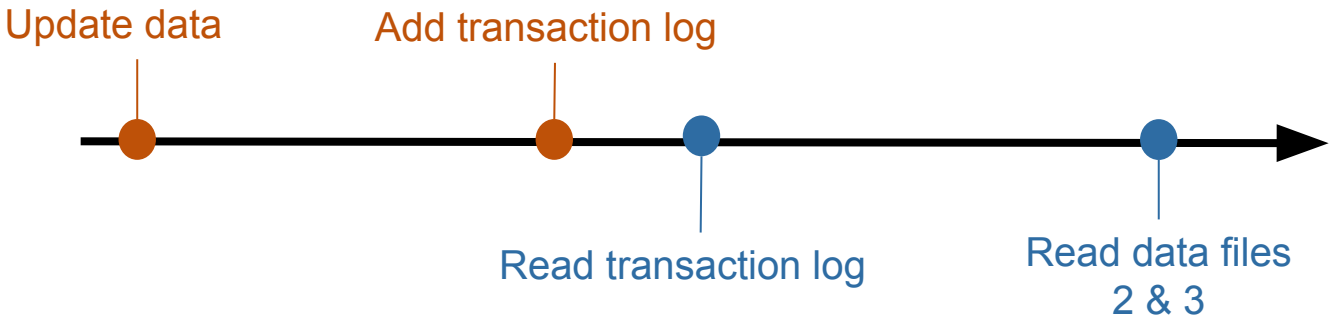
_delta_log

00.json

Updates

Writer

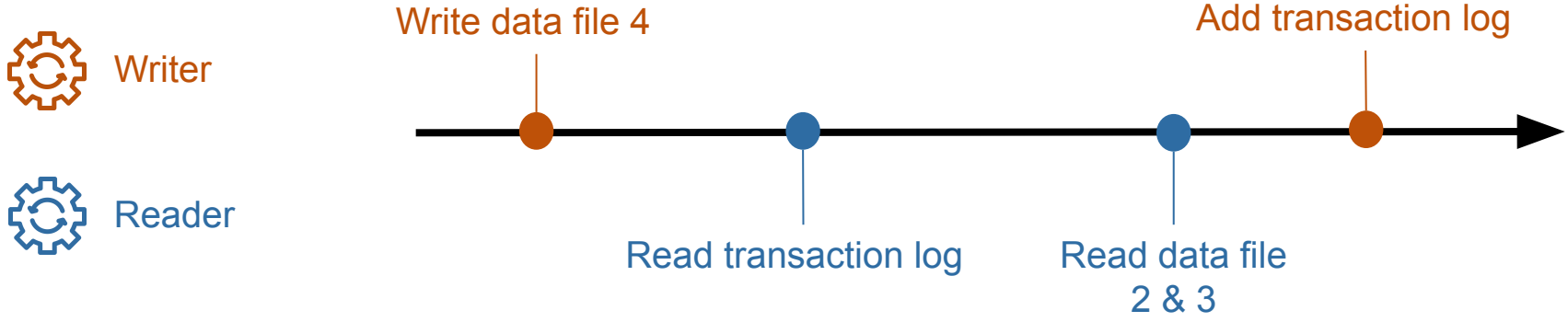
Reader



Data Files

_delta_log

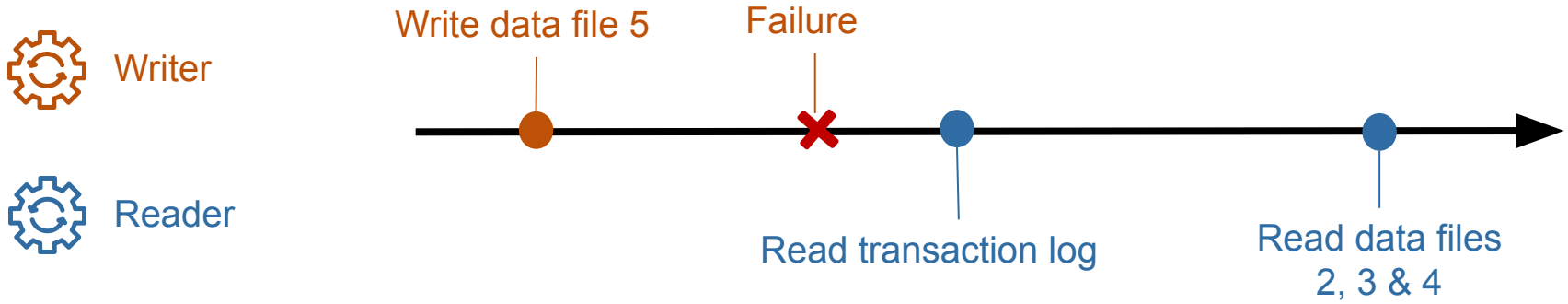
Simultaneous Operations



Data Files

_delta_log

Failed Writes



Delta Lake Advantages

- Brings ACID transactions to object storage
- Handle scalable metadata
- Full audit trail of all changes
- Builds upon standard data formats: Parquet + Json

Presentation

- 1.2 - Understanding Delta Tables

Hands-on Lab

In your Databricks workspace:

- Solve Lab: **1.2L - Delta Lake**

Estimated Time: 15 Minutes

- Create your Databricks workspace in your own cloud account

Q&A

Advanced Delta Lake Features

Learning Objectives

- Time travel
- Compacting Small Files and Indexing
- Vacuum

Time travel

- Audit data changes
- **DESCRIBE HISTORY** command

Time travel

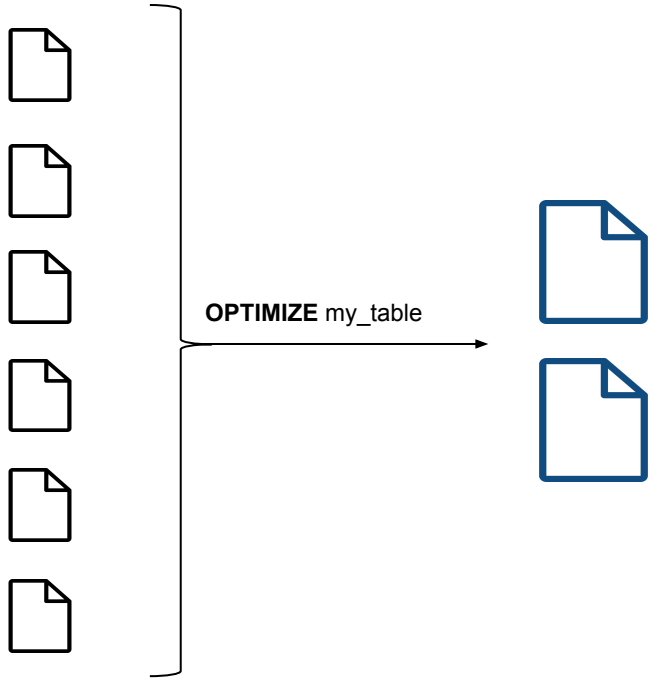
- Query older versions of the data
- Using a timestamp
  - SELECT * FROM my_table **TIMESTAMP AS OF** "2019-01-01"
- Using a version number
  - SELECT * FROM my_table **VERSION AS OF** 36
  - SELECT * FROM my_table**@v**36

Time travel

- Rollback Versions
- **RESTORE TABLE** command:
  - **RESTORE TABLE** my_table **TO TIMESTAMP AS OF** "2019-01-01”
  - **RESTORE TABLE** my_table **TO VERSION AS OF** 36

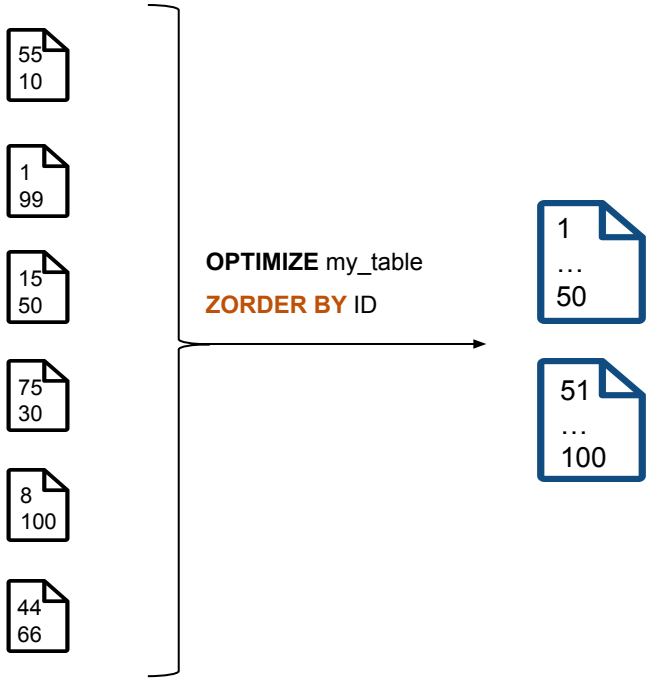
Compaction

- Compacting Small Files
- **OPTIMIZE** command:
  - **OPTIMIZE** my_table



Indexing

- Co-locate column information
- **OPTIMIZE** command:
  - OPTIMIZE my_table
    ZORDER BY column_name



Vacuum a Delta table

- Cleaning up unused data files
  - uncommitted files
  - files that are no longer in in latest table state

- **VACUUM** command
  - **VACUUM** table_name [retention period]
  - Default retention period: 7 days

- **Note: Vacuum = no time travel**

Presentation

- 1.3 - Advanced Delta Lake Features

Poll

- Which of the following commands can a data engineer use to compact small data files of a Delta table into larger ones ?

A. RESTORE TABLE

B. ZORDER BY

C. COMPACT

D. VACUUM

E. OPTIMIZE

Poll

- Which of the following commands can a data engineer use to compact small data files of a Delta table into larger ones ?

A. RESTORE TABLE

B. ZORDER BY

C. COMPACT

D. VACUUM

E. OPTIMIZE

Q&A

Relational Entities on Databricks

Learning Objectives

- Databases
- Tables
- The impact of the LOCATION keyword

Database

- Databases = Schemas in Hive metastore

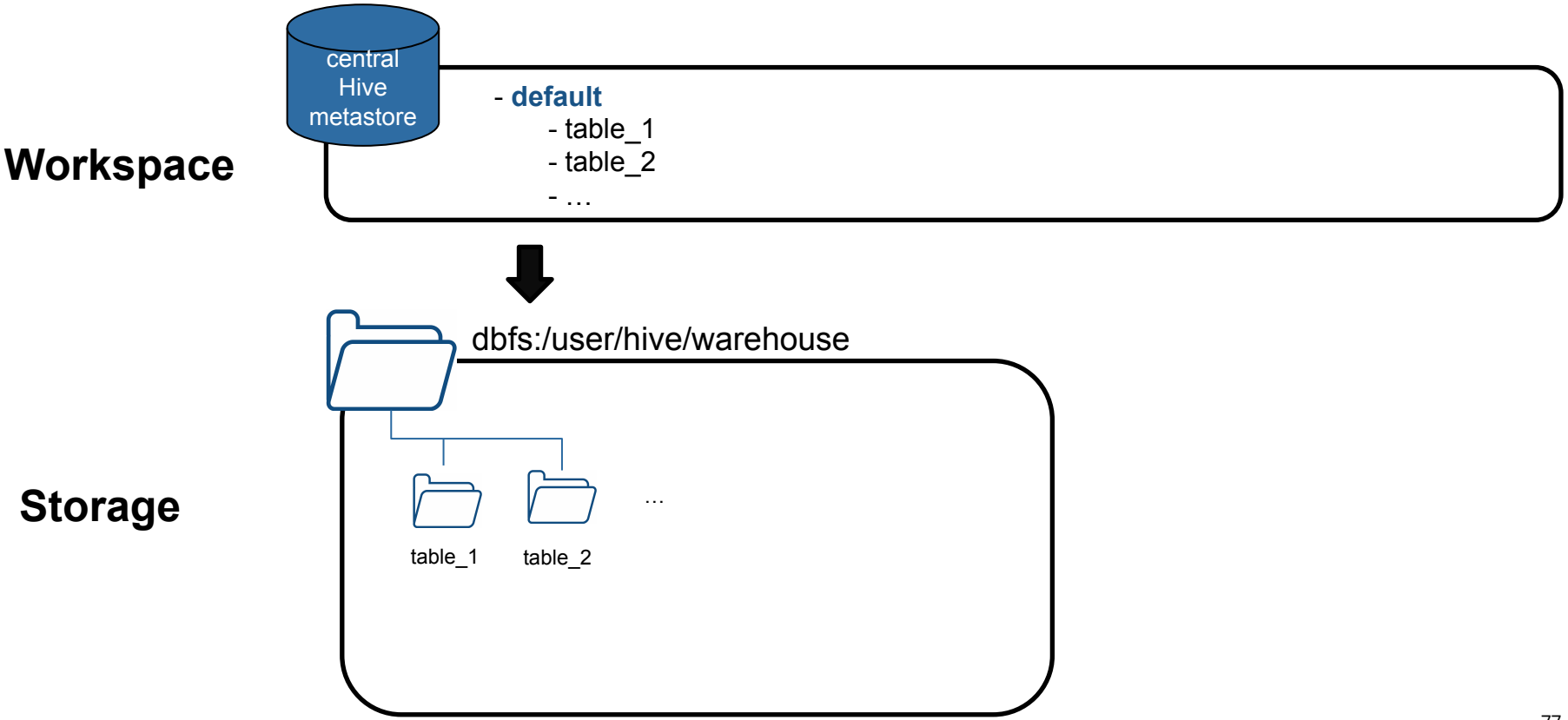
- CREATE DATABASE db_name
- CREATE SCHEMA db_name

Hive metastore

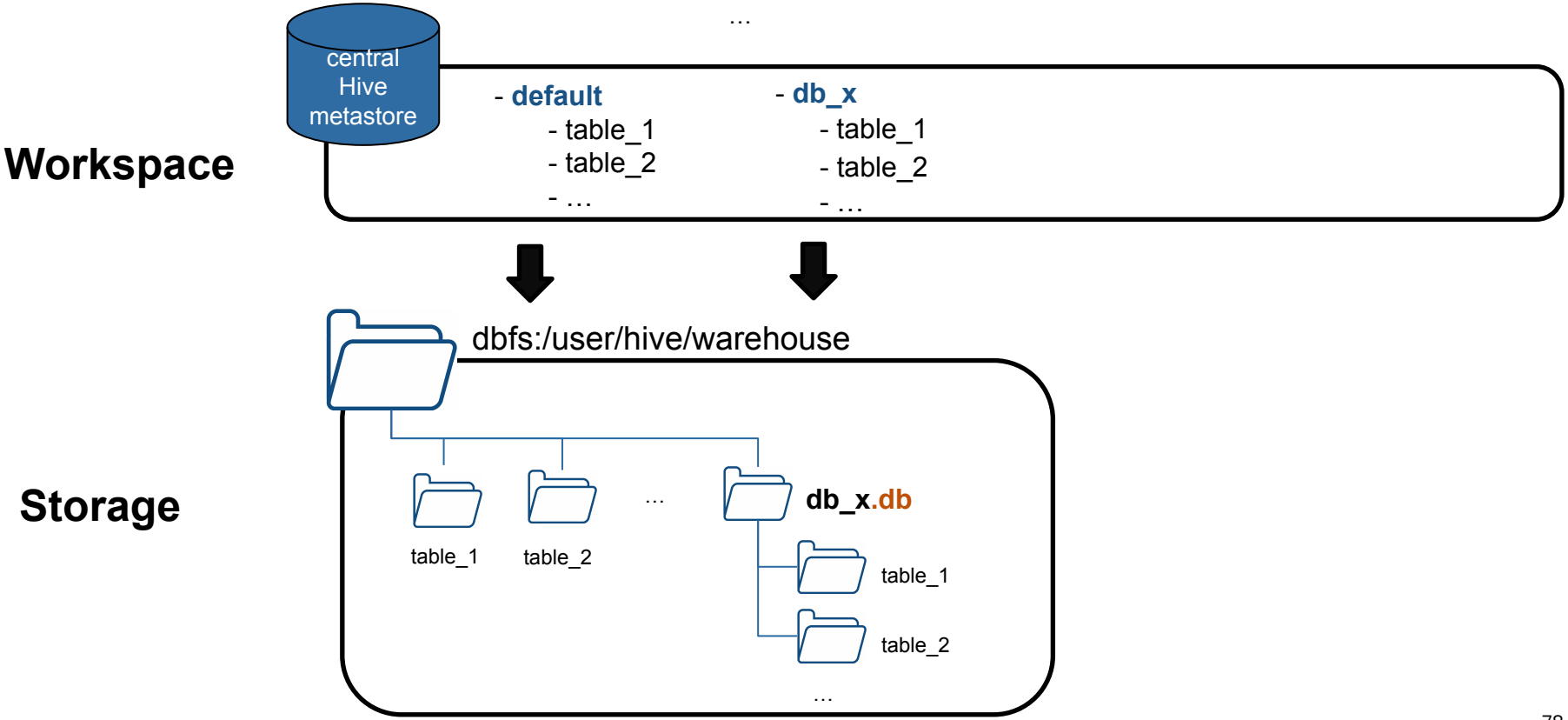
- repository of metadata
  - Databases
  - Tables
  - …

CREATE TABLE table1;

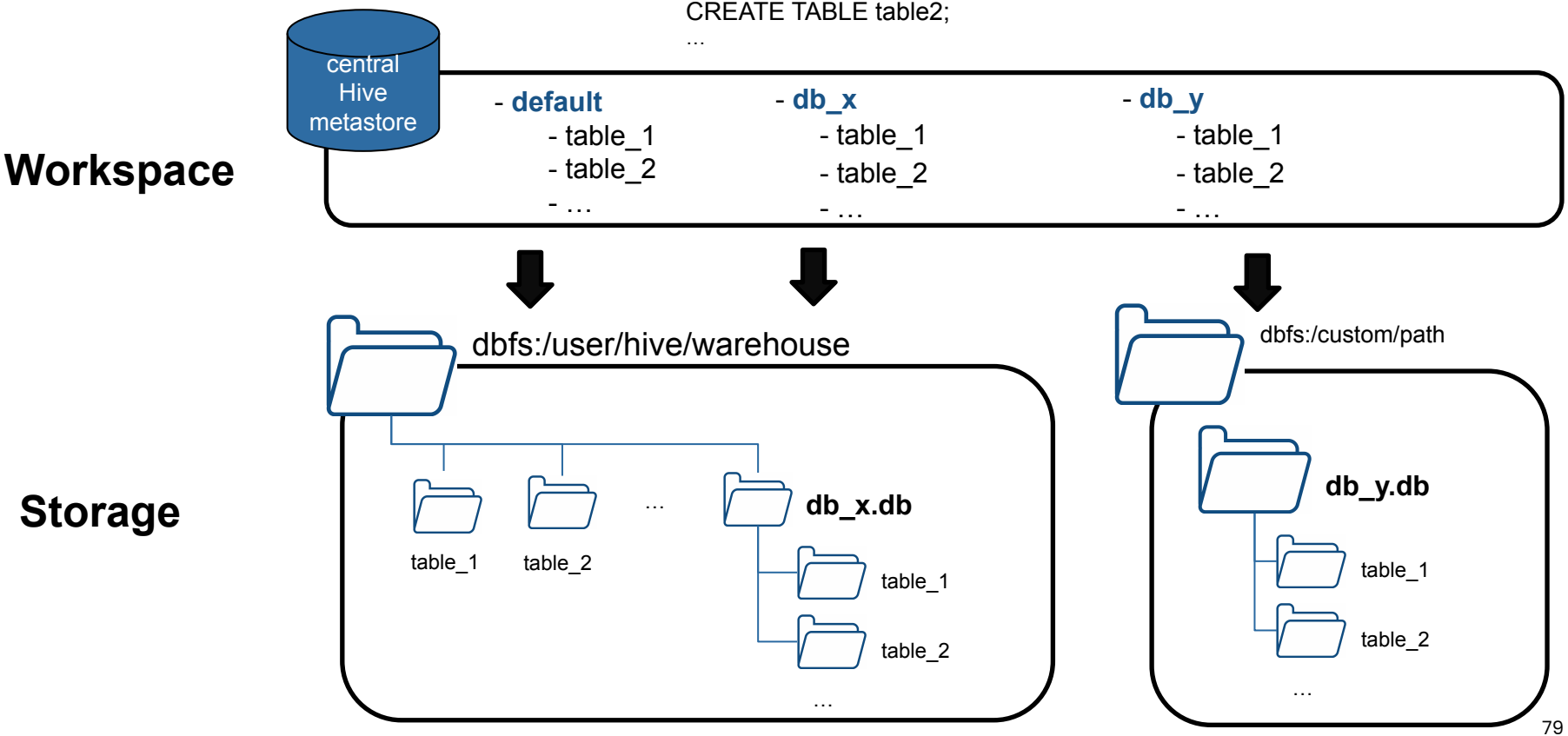
CREATE TABLE table2;
…



```
CREATE SCHEMA db_x

USE db_x;

CREATE TABLE table1;
CREATE TABLE table2;
…
```



```
CREATE SCHEMA db_y
LOCATION 'dbfs:/custom/path/db_y.db'

USE db_y;
CREATE TABLE table1;
CREATE TABLE table2;
...
```



Tables

Manged tables

Created under the database directory
CREATE TABLE table_name

Dropping the table, delete the underlying data files

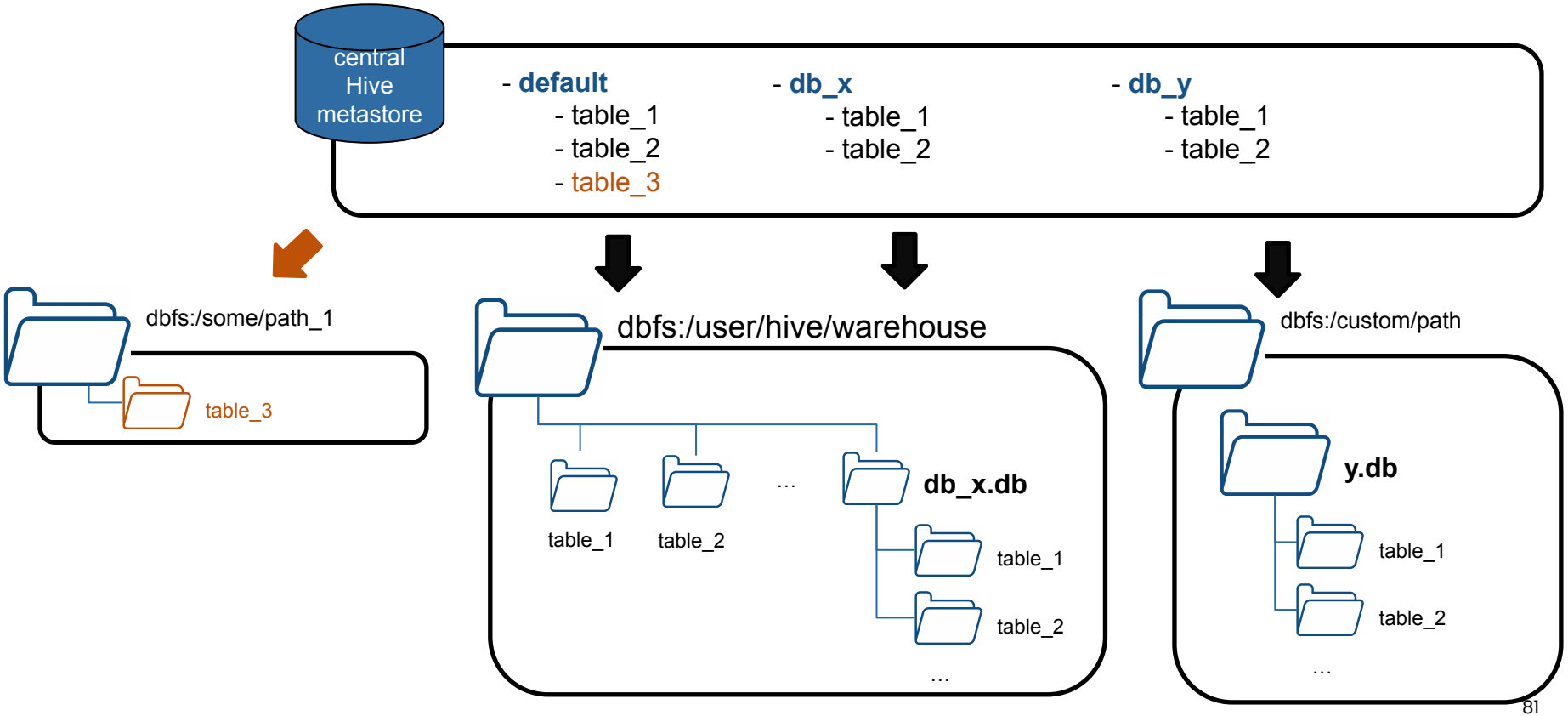
External tables

Created outside the database directory
CREATE TABLE table_name
LOCATION ‘path’

Dropping the table, will **Not** delete the underlying data files

CREATE TABLE table3

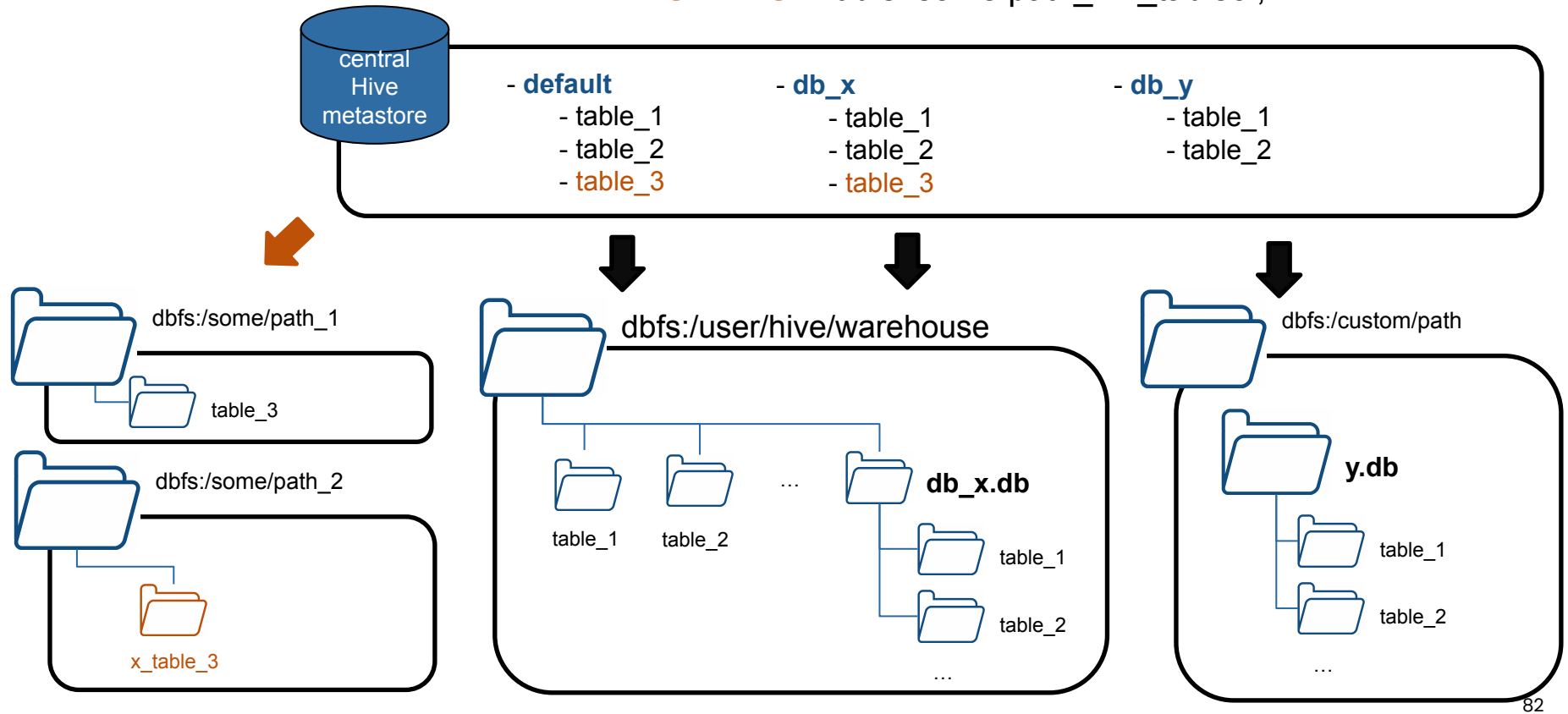
LOCATION ‘dbfs:/some/path_1/table3’



USE db_x;

CREATE TABLE table3

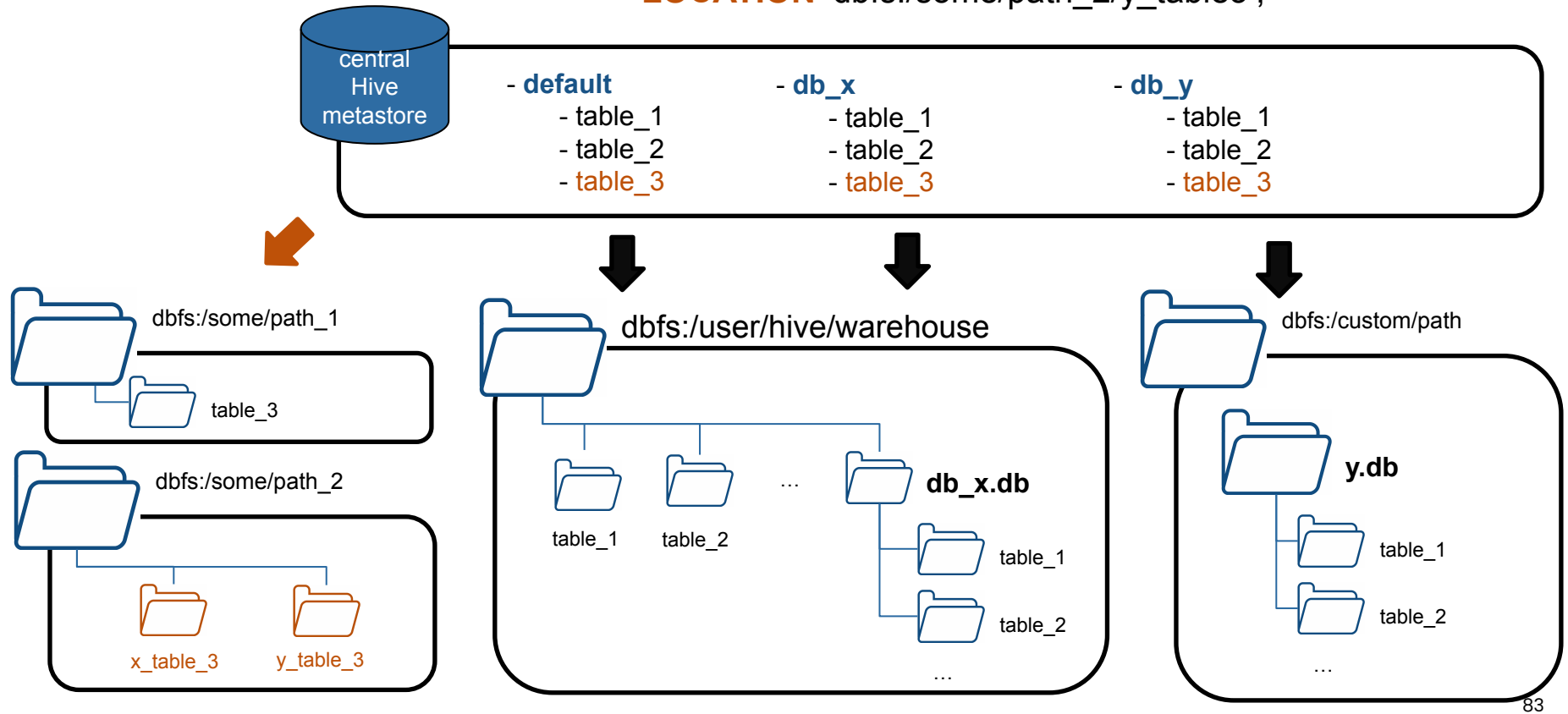
LOCATION ‘dbfs:/some/path_2/x_table3’;



USE db_y;

CREATE TABLE table3

LOCATION ‘dbfs:/some/path_2/y_table3’;



Presentation

- 1.4 - Databases and Tables on Databricks

Hands-on Lab

In your Databricks workspace:

- Solve Lab: **1.3L - Databases and Tables on Databricks**

Estimated Time: 20 Minutes

- Create your Databricks workspace in your own cloud account

Q&A

Set Up Delta Tables

Learning Objectives

- CTAS statements
- Table Constraints
- Cloning Delta Lake Tables

CTAS

- CREATE TABLE _ AS SELECT statement
- **CREATE TABLE** table_1
  AS SELECT * FROM table_2
- Automatically infer schema information from query results
  - do not support manual schema declaration

CTAS: Filtering and Renaming Columns

- **CREATE TABLE** table_1

  AS SELECT col_1, col_3 AS new_col_3 FROM table_2

CTAS: Additional Options

- CREATE TABLE new_table

  COMMENT "Contains PII"

  PARTITIONED BY (city, birth_date)

  LOCATION '/some/path'

  AS SELECT id, name, email, birth_date, city FROM users

CREATE TABLE vs. CTAS

CREATE TABLE

CREATE TABLE table_1
(col1 INT, col2 STRING, col3 DOUBLE)

Manual schema declaration

Create empty table
Need **INSERT INTO** statement

CTAS

CREATE TABLE table_1
AS SELECT col1, col2, col3 FROM table_2

Do **Not** support manual schema declaration
Automatically infer schema

Table created with data

Table Constraints

- NOT NULL constraints
- CHECK constraints

- **ALTER TABLE** table_name **ADD CONSTRAINT** constraint_name constraint_details

- **ALTER TABLE** orders **ADD CONSTRAINT** valid_date **CHECK** (date > '2020-01-01');

Cloning Delta Lake Tables

- DEEP CLONE
- SHALLOW CLONE

Deep Cloning

- Fully copies data + metadata from a source table to a target
- **CREATE TABLE** table_clone
  DEEP CLONE source_table
- Can sync changes
- Take quite a while for large datasets

Shallow Cloning

- Quickly create a copy of a table
  - Just copy the Delta transaction logs
- **CREATE TABLE** table_clone
  SHALLOW CLONE source_table

Cloning Delta Lake Tables

- Useful to set up tables for testing in development.
- In either case, data modifications will not affect the source

Poll

- Which of the following statements is ***Not*** true about CTAS statements ?

A. CTAS statements automatically infer schema information from query results

B. CTAS statements support manual schema declaration

C. CTAS statements stand for CREATE TABLE _ AS SELECT statement

D. With CTAS statements, data will be inserted during the table creation

E. All these statements are Not true about CTAS statements

Poll

- Which of the following statements is ***Not*** true about CTAS statements ?

A. CTAS statements automatically infer schema information from query results

B. CTAS statements support manual schema declaration

C. CTAS statements stand for CREATE TABLE _ AS SELECT statement

D. With CTAS statements, data will be inserted during the table creation

E. All these statements are Not true about CTAS statements

Q&A

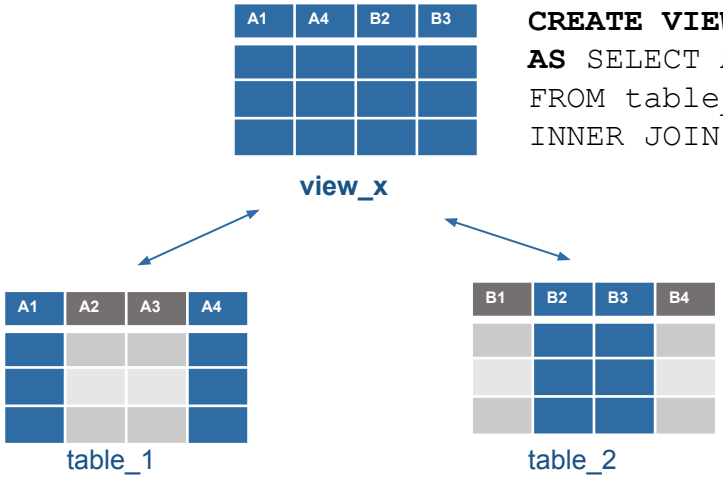
Views

Learning Objectives

- Views
- Types of views

View

- Logical query against source tables




```
CREATE VIEW view_x
AS SELECT A1, A4, B2, B3
FROM table_1
INNER JOIN table_2 …
```

Types of views

1. (Stored) Views
2. Temporary views
3. Global Temporary views

1- (Stored) Views

- Persisted objects
- **CREATE VIEW** view_name
  AS query

2- Temporary view

- Session-scoped view
- **CREATE TEMP VIEW** view_name
  AS query

Spark Session

- Opening a new notebook
- Detaching and reattaching to a cluster
- Installing a python package
- Restarting a cluster

3- Global Temporary views

- Cluster-scoped view
- **CREATE GLOBAL TEMP VIEW** view_name
  AS query
- SELECT * FROM **global_temp**.view_name

Views comparison

(Stored) Views	Temp views	Global Temp views
Persisted in DB	Session-scoped	Cluster-scoped
Dropped only by **DROP VIEW**	dropped when session ends	dropped when cluster restarted
CREATE VIEW	CREATE **TEMP** VIEW	CREATE **GLOBAL TEMP** VIEW

Presentation

- 1.5A – Views
- 1.5B - Views (Session 2)

Poll

- A data engineer wants to create a relational object by pulling data from two tables. The relational object will only be used in the current session. In order to save on storage costs, the date engineer wants to avoid copying and storing physical data.

  Which of the following relational objects should the data engineer create?

  A. External table

  B. Temporary view

  C. Managed table

  D. Global Temporary view

  E. View

Poll

- A data engineer wants to create a relational object by pulling data from two tables. The relational object will only be used in the current session. In order to save on storage costs, the date engineer wants to avoid copying and storing physical data.

  Which of the following relational objects should the data engineer create?

  A. External table

  B. Temporary view

  C. Managed table

  D. Global Temporary view

  E. View

Q&A

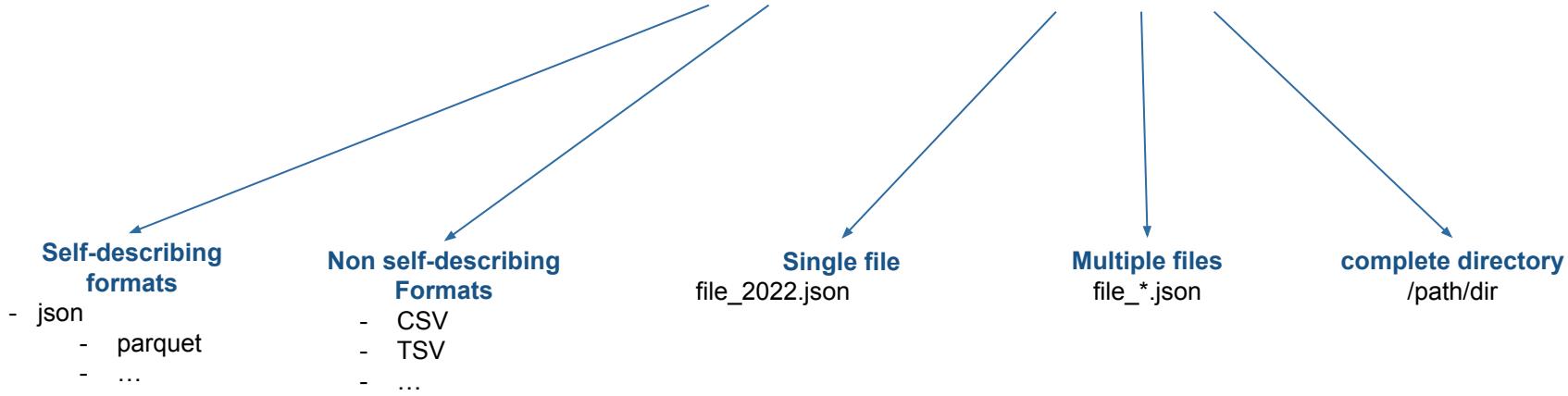
Querying Files

Learning Objectives

- Querying data files directly
- Extract files as raw contents
- Configure options of external sources
- Use CTAS statements to create Delta Lake tables

Querying Files Directly

SELECT * FROM file_format.`/path/to/file`



Example: JSON

```
SELECT * FROM json.`/path/file_name.json`
```

Raw data

- Extract text files as raw strings
  - Text-based files (JSON, CSV, TSV, and TXT formats)
  - SELECT * FROM **text**.`/path/to/file`
- Extract files as raw bytes
  - Images or unstructured data
  - SELECT * FROM **binaryFile**.`/path/to/file`

CTAS: Registering Tables from Files

- **CREATE TABLE** table_name

  AS SELECT * FROM file_format.`/path/to/file`

- Automatically infer schema information from query results
  - Do **Not** support manual schema declaration.
  - Useful for external data ingestion with well-defined schema

- Do **Not** support file options

Registering Tables on External Data Sources

- **CREATE TABLE** table_name
  (col_name1 col_type1, ...)
  USING data_source
  OPTIONS (key1 = val1, key2 = val2, ...)
  LOCATION = path

- External table

- Non-Delta table!

Example: CSV

- **CREATE TABLE** table_name
  (col_name1 col_type1, ...)
  USING CSV
  OPTIONS (header = "true",
  delimiter = ";")
  LOCATION = path

Example: Database

- **CREATE TABLE** table_name
  (col_name1 col_type1, ...)
  USING JDBC
  OPTIONS (url = "jdbc:sqlite://hostname:port",
  dbtable = "database.table",
  user = "username",
  password = ”pwd”)

Limitation

- It’s Not Delta table!
- We can not expect the performance guarantees associated with Delta Lake and Lakehouse
- Having a huge database table

Solution

- **CREATE TEMP VIEW** temp_view_name (col_name1 col_type1, ...)
  USING data_source
  OPTIONS (key1 = “val1”, key2 = “val2”, ..., path = “/path/to/files”)

- **CREATE TABLE** table_name
  AS SELECT * FROM temp_view_name

Presentation

- 2.1 - Querying Files
- 2.2 - Writing to Tables

Hands-on Lab

In your Databricks workspace:

- Solve Lab: **2.1L - Querying Files**

Estimated Time: 20 Minutes

- Create your Databricks workspace in your own cloud account

Q&A

Tomorrow …

- Advanced ETL
- Spark Structured Streaming
- Incremental Data Ingestion from Files
- Multi-hop architecture
- Delta Live Tables (DLT)

O'REILLY®

Databricks Data Engineer Associate Certification Prep in 2 Weeks

Day 3

Derar Alhussein

Day 3 Agenda

- Advanced ETL
- Spark Structured Streaming
- Incremental Data Ingestion from Files
- Multi-hop architecture
- Delta Live Tables (DLT)

Advanced ETL

Presentation

- 2.3 - Advanced Transformations
- 2.4 - Higher Order Functions and SQL UDFs

Hands-on Lab

In your Databricks workspace:

- Solve Lab: **2.2L - Advanced ETL**

Estimated Time: 20 Minutes

- Create your Databricks workspace in your own cloud account

Q&A

Structured Streaming

Learning Objectives

- Process streaming data
- DataStreamReader
- DataStreamWriter

Data Stream

- Any data source that grows over time

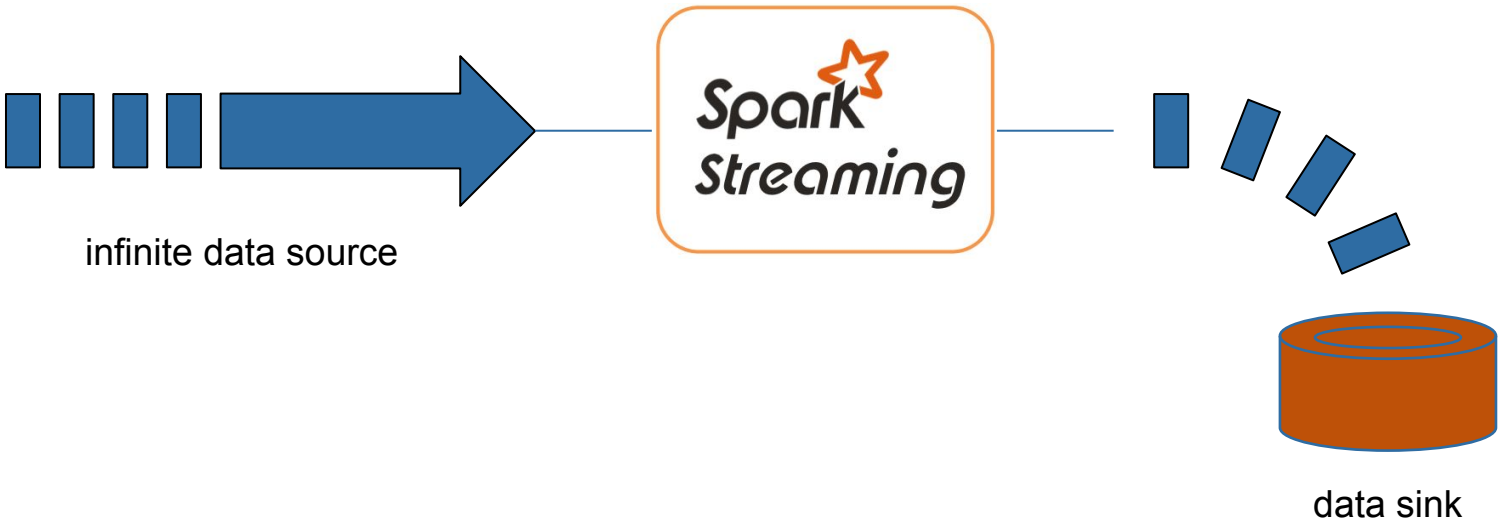
- New files landing in cloud storage
- Updates to a database captured in a CDC feed
- Events queued in a pub/sub messaging feed

Processing Data Stream

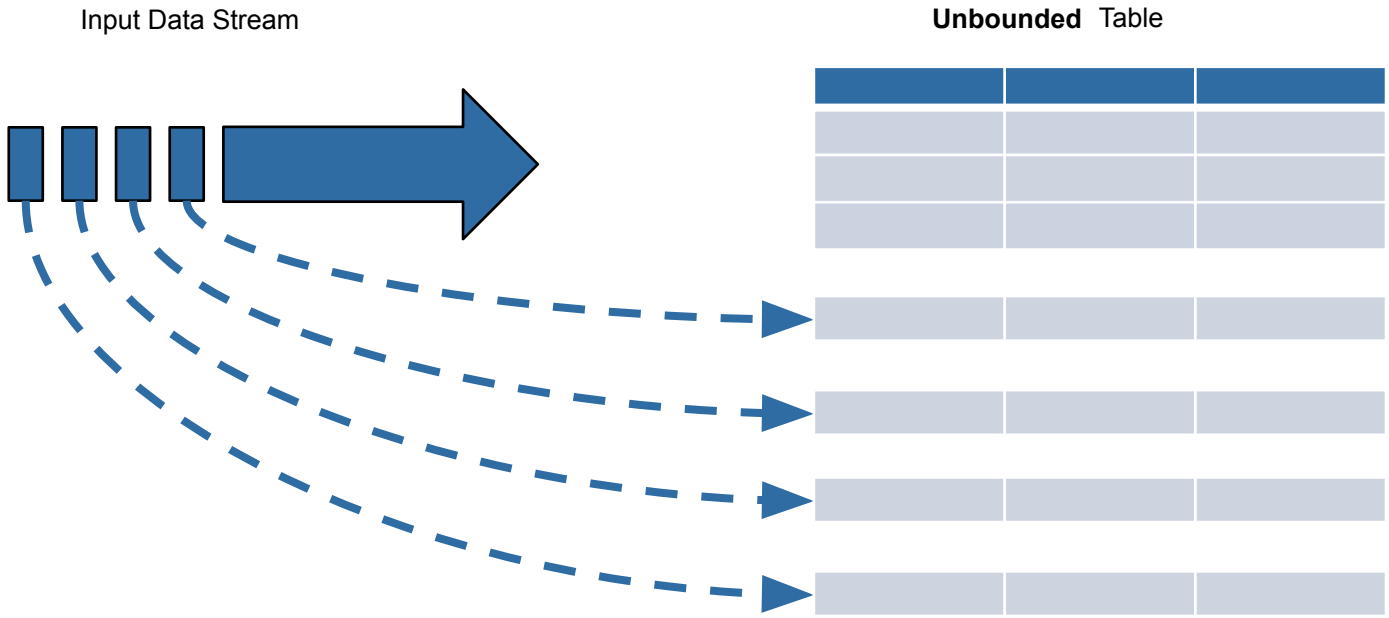
2 approaches:

1. Reprocess the entire source dataset each time
2. Only process those new data added since last update
    - Structured Streaming

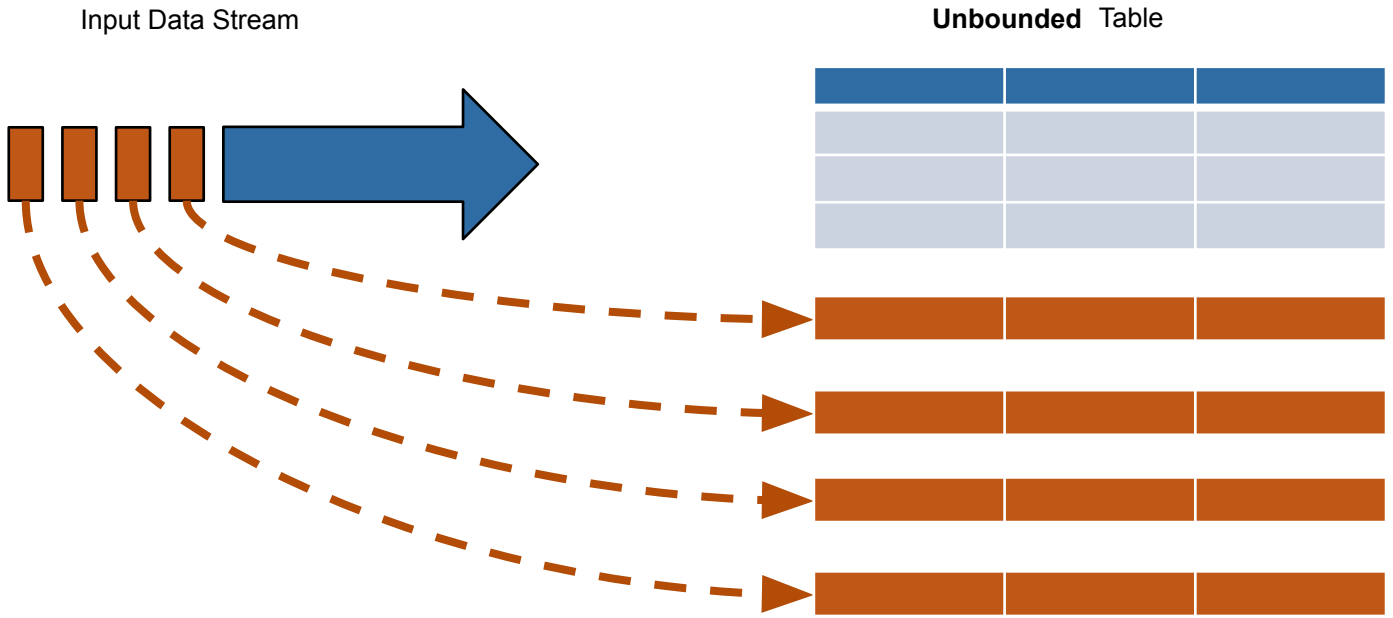
Spark Structured Streaming



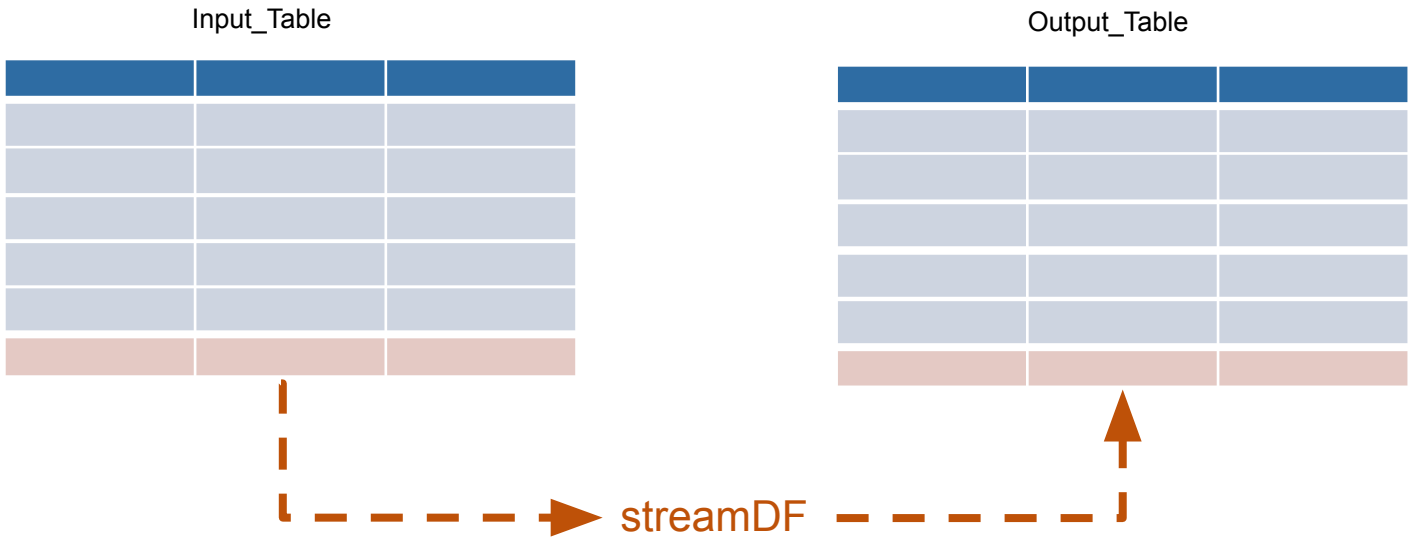
Treating Infinite Data as a Table



Append-only requirement



Input Streaming Table




```
streamDF = spark.readStream
                .table("Input_Table")
```

```
streamDF.writeStream
        .trigger(processingTime="2 minutes")
        .outputMode("append")
        .option("checkpointLocation", "/path")
        .table("Output_Table")
```

Trigger Intervals

```
streamDF.writeStream
      .trigger(processingTime="2 minutes")
      .outputMode("append")
      .option("checkpointLocation", "/path")
      .table("Output_Table")
```

Trigger	Method call	Behavior
Unspecified		Default: processingTime="500ms"
Fixed interval	`.trigger(processingTime="5 minutes")`	Process data in micro-batches at the user-specified intervals
Triggered batch	`.trigger(once=True)`	Process all available data in a Single batch, then stop
Triggered micro-batches	`.trigger(availableNow=True)`	Process all available data in multiple micro-batches, then stop

Output Modes

```
streamDF.writeStream
     .trigger(processingTime="2 minutes")
     .outputMode("append")
     .option("checkpointLocation", "/path")
     .table(”Output_Table")
```

Mode	Method call	Behavior
Append (Default)	`.outputMode("append")`	Only newly appended rows are incrementally appended to the target table with each batch
Complete	`.outputMode("complete")`	The target table is overwritten with each batch

Checkpointing

```
streamDF.writeStream
     .trigger(processingTime="2 minutes")
     .outputMode("append")
     .option("checkpointLocation", "/path")
     .table(”Output_Table")
```

- Store stream state
- Track the progress of your stream processing

- Can **Not** be shared between separate streams

Guarantees

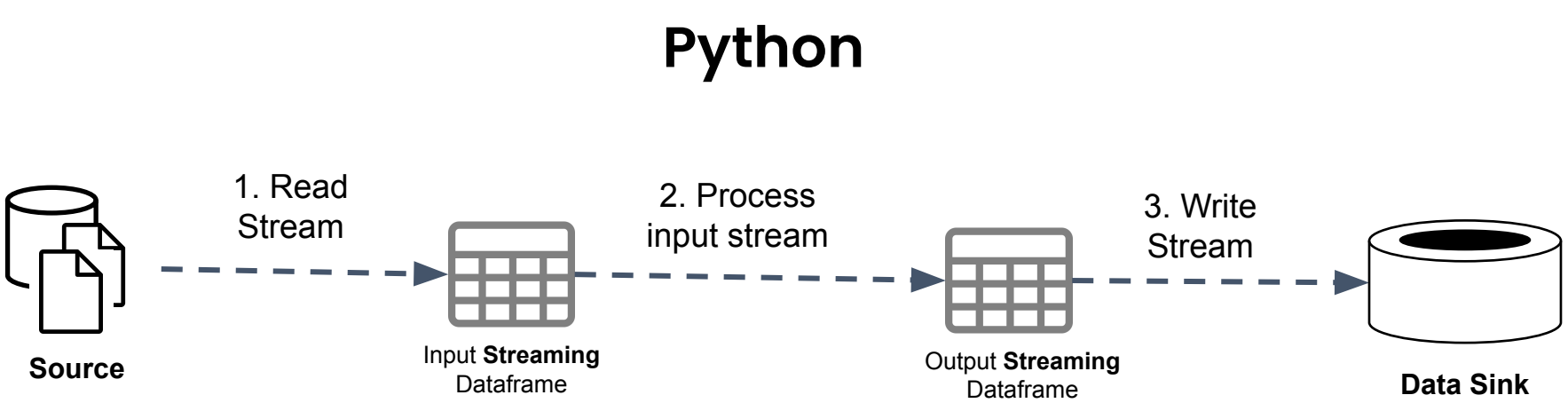
1. Fault Tolerance
    - Checkpointing + Write-ahead logs
        - record the offset range of data being processed during each trigger interval.

2. Exactly-once guarantee
    - Idempotent sinks

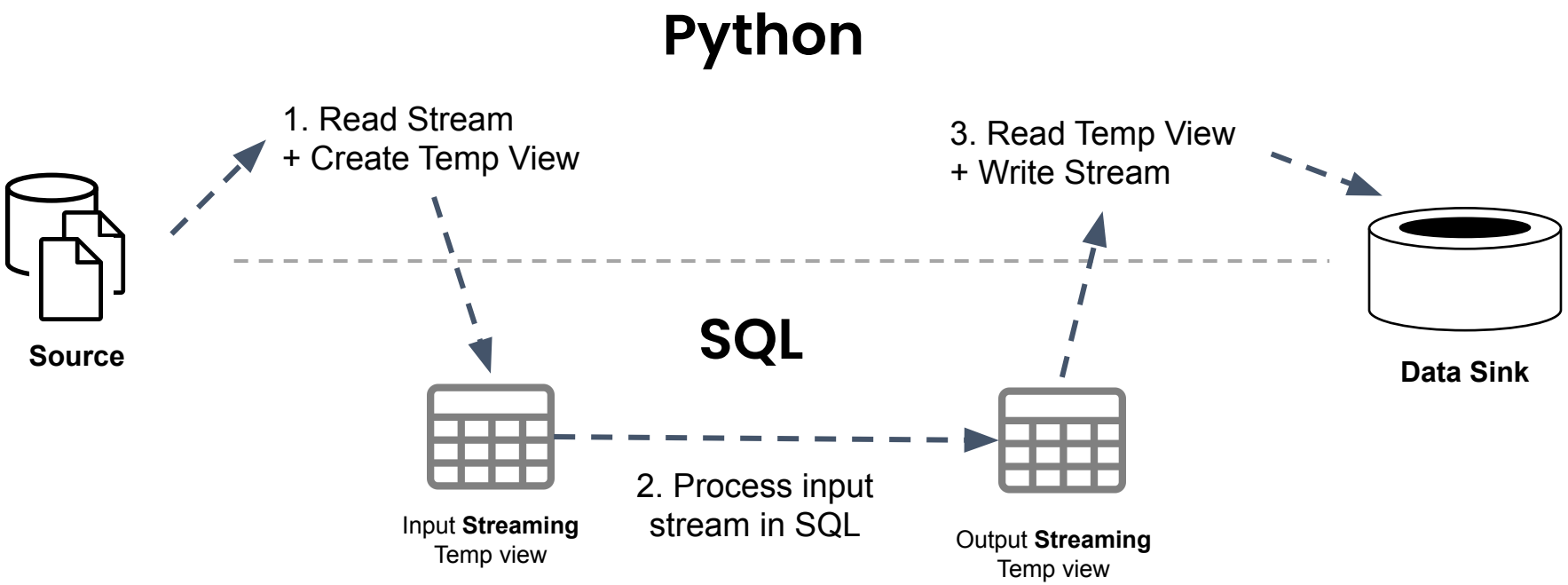
Unsupported Operations

- Some operations are not supported by streaming DataFrame
  - Sorting
  - Deduplication
- Advanced methods
  - Windowing
  - Watermarking

Streaming in Python



Streaming in SQL



Presentation

- 3.1 - Structured Streaming

Hands-on Lab

In your Databricks workspace:

- Solve Lab: **3.1L - Spark Structured Streaming**

Estimated Time: 25 Minutes

- Create your Databricks workspace in your own cloud account

Q&A

Incremental Data Ingestion from Files

Learning Objectives

- What is incremental data Ingestion from file
- COPY INTO
- Auto Loader

Incremental Data Ingestion

- Loading new data files encountered since the last ingestion
- Reduces redundant processing
- 2 mechanisms:
  - COPY INTO
  - Auto loader

COPY INTO

- SQL command
- Idempotently and incrementally load new data files
  - Files that have already been loaded are skipped.

COPY INTO

- **COPY INTO** my_table
  FROM '/path/to/files'
  FILEFORMAT = <format>
  FORMAT_OPTIONS (<format options>)
  COPY_OPTIONS (<copy options>);

Example

- **COPY INTO** my_table
  FROM '/path/to/files’
  FILEFORMAT = CSV
  FORMAT_OPTIONS ('delimiter' = '|’,
  'header' = 'true')
  COPY_OPTIONS ('mergeSchema' = 'true’)

Auto loader

- Structured Streaming
- Can process billions of files
- Support near real-time ingestion of millions of files per hour.

Auto loader Checkpointing

- Store metadata of the discovered files
- Exactly-once guarantees
- Fault tolerance

Auto Loader in PySpark API

```
spark.readStream
        .format("cloudFiles")
        .option("cloudFiles.format", <source_format>)
        .load('/path/to/files')
    .writeStream
        .option("checkpointLocation", <checkpoint_directory>)
        .table(<table_name>)
```

Auto Loader + Schema

```
spark.readStream
        .format("cloudFiles")
        .option("cloudFiles.format", <source_format>)
                .option("cloudFiles.schemaLocation", <schema_directory>)
        .load('/path/to/files')
    .writeStream
        .option("checkpointLocation", <checkpoint_directory>)
                .option("mergeSchema", “true”)
        .table(<table_name>)
```

COPY INTO vs. Auto Loader

COPY INTO	Auto Loader
Thousands of files	Millions of files
Less efficient at scale	Efficient at scale

Presentation

- 3.2 - Auto Loader

Poll

- Which of the following techniques allows Auto Loader to track the ingestion progress and store metadata of the discovered files ?

A. DEEP CLONE

B. COPY INTO

C. Watermarking

D. Checkpointing

E. Z-Ordering

Poll

- Which of the following techniques allows Auto Loader to track the ingestion progress and store metadata of the discovered files ?

A. DEEP CLONE

B. COPY INTO

C. Watermarking

D. Checkpointing

E. Z-Ordering

Q&A

Multi-Hop Architecture

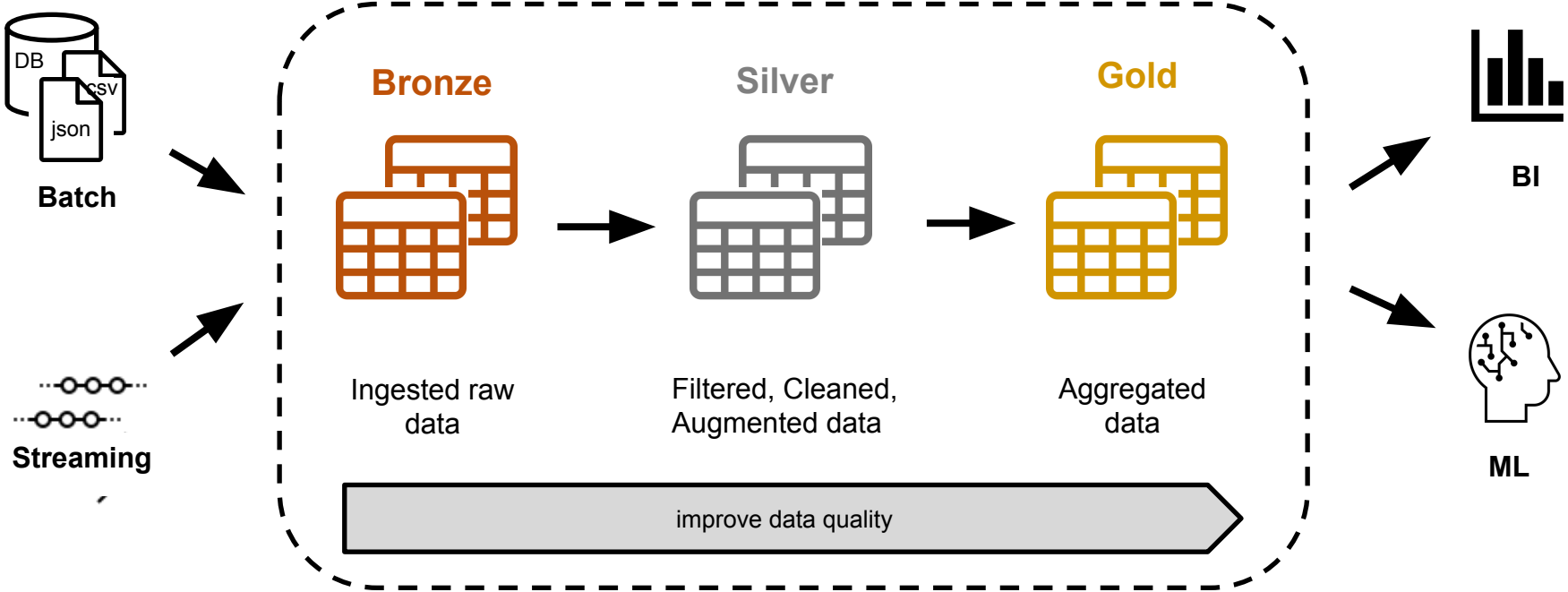
Learning Objectives

- Incremental Multi-Hop pipeline
- Describe Bronze, Silver, and Gold tables

Multi-Hop Architecture

- Medallion Architecture
- Organize data in multi-layered approach
- Incrementally improving the structure and quality of data as it flows through each layer

Multi-Hop Architecture



Benefits

- Simple data model
- Enables incremental ETL
- Combine streaming and batch workloads in unified pipeline
- Can recreate your tables from raw data at any time

Presentation

- 3.3 - Multi-Hop Architecture

Hands-on Lab

In your Databricks workspace:

- Solve Lab: **3.2L - Multi-Hop Architecture**

Estimated Time: 30 Minutes

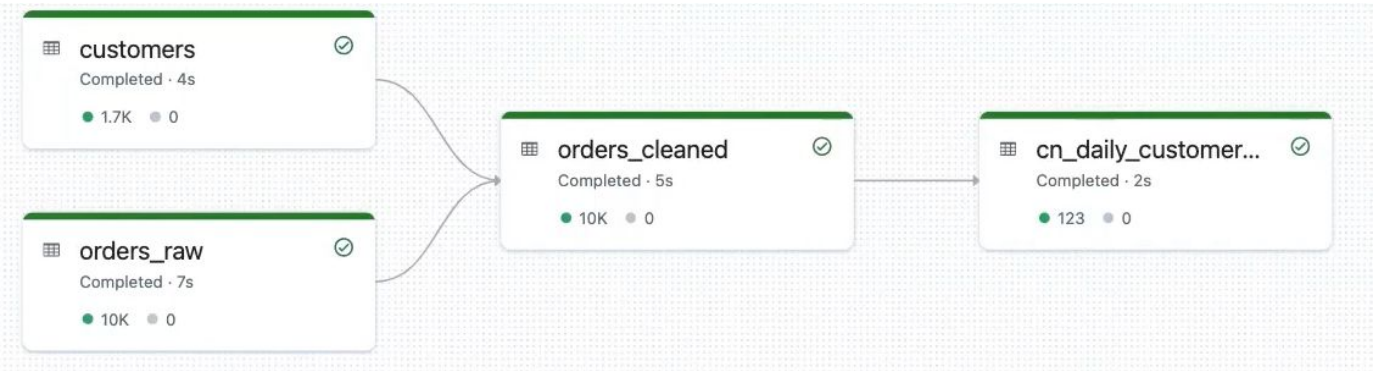
- Create your Databricks workspace in your own cloud account

Q&A

Delta Live Tables (DLT)

Delta Live Tables

- Delta Live Tables, or DLT.
- Framework for building reliable and maintainable data processing pipelines.



Benefits

- DLT simplify the hard work of building large scale ETL
- Maintain table dependencies
- Support data quality control

DLT vs. Spark Structured Streaming

Spark Structured Streaming

```
spark.readStream
        .format("cloudFiles")
        .option("cloudFiles.format", "json")
        .load('/some/path/')
    .writeStream
        .option("checkpointLocation", "/path")
        .table("orders_raw")
```

DLT

```
import dlt

@dlt.table
def orders_raw():
    return (spark.readStream
            .format("cloudFiles")
            .option("cloudFiles.format", "json")
            .load("/some/path")
        )
```

DLT vs. Spark Structured Streaming

Spark Structured Streaming

Can Not create streaming tables in Spark SQL syntax only. Need to pass by PySpark to register streaming tables

Does Not support data quality control

DLT

Support creating streaming tables in SQL via **CREATE STREAMING TABLE** syntax

Ensure data quality using DLT Expectations

DLT Object types

- Streaming Tables
- Live Tables (Materialized View)
- Live Views

Streaming Tables vs. Live Tables

Streaming Tables

Process data added only since the last pipeline run

CREATE STREAMING [LIVE] TABLE _ AS syntax

Input data source must be a streaming source:

- Autoloader
- Incremental (append-only) table read via **STREAM()** function

Live Tables (Materialized View)

Reprocess the entire source dataset with each pipeline run

CREATE LIVE TABLE _ AS syntax

Input data dataset is static, or contains data updated, deleted, or overwritten

Live Views

- Temporary views scoped to the DLT pipeline they are a part of
  - Not persisted to the catalog
- Used for Intermediate transformations and data quality checks
- **CREATE LIVE VIEW _ AS** syntax

Presentation

- 4.1 - Delta Live Tables
- 4.2 - Pipeline Results

Hands-on Lab

In your Databricks workspace:

- Solve Lab: **4.1L - Delta Live Tables**

Estimated Time: 40 Minutes

- Create your Databricks workspace in your own cloud account

Q&A

Tomorrow ...

- Change Data Capture (CDC)
- Databricks Jobs
- Databricks SQL
- Data objects privileges
- Unity Catalog

- Certification Overview

O'REILLY®

Databricks Data Engineer Associate Certification Prep in 2 Weeks

Day 4

Derar Alhussein

Day 4 Agenda

- Change Data Capture (CDC)
- Databricks Jobs
- Databricks SQL
- Data objects privileges
- Unity Catalog

- Certification Overview

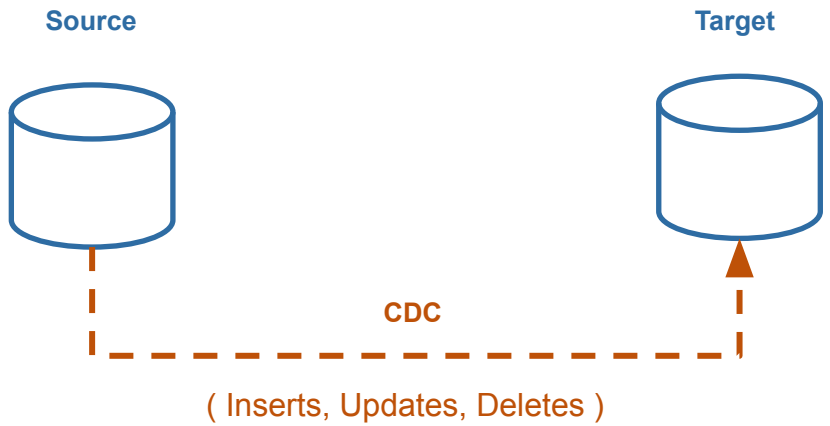
Change Data Capture

Learning Objectives

- Definition of Change data capture
- Processing Change data capture with DLT

Change Data Capture

- CDC: Process of identifying changes made to data in the source and delivering those changes to the target



Row-Level changes

- Inserting new records
- Updating existing records
- Deleteing existing records

CDC Feed

- Row data + metadata

Country ID	Country	Vaccination Rate	sequence	operation
FR	France	0.74	2022-11-01 07:00	UPDATE
FR	France	0.75	2022-11-01 08:00	UPDATE
CA			2022-11-01 00:00	DELETE
US	USA	0,5	2022-11-01 00:00	INSERT
IN	India	0.66	2022-11-01 00:00	INSERT

CDC Feed

Country ID	Country	Vaccination Rate
FR	France	0,6
CA	Canada	0,71

Target table

CDC Feed

- Row data + metadata

Country ID	Country	Vaccination Rate	sequence	operation
FR	France	0.74	2022-11-01 07:00	UPDATE
FR	France	0.75	2022-11-01 08:00	UPDATE
CA			2022-11-01 00:00	DELETE
US	USA	0.5	2022-11-01 00:00	INSERT
IN	India	0.66	2022-11-01 00:00	INSERT

CDC Feed

Country ID	Country	Vaccination Rate
FR	France	0.75
US	USA	0.5
IN	India	0.66

Target table

CDC with DLT

- APPLY CHANGES INTO command

```
APPLY CHANGES INTO LIVE.target_table
FROM STREAM(LIVE.cdc_feed_table)
KEYS (key_field)
APPLY AS DELETE WHEN operation_field = "DELETE"
SEQUENCE BY sequence_field
COLUMNS *
```

APPLY CHANGES INTO: Pros

- Orders late-arriving records using the sequencing key
- Default assumption is that rows will contain inserts and updates
- Can optionally apply deletes (APPLY AS DELETE WHEN condition)
- Specify one or many fields as the primary key for a table
- Specify columns to ignore with the EXCEPT keyword

APPLY CHANGES INTO: **Cons**

- Breaks the append-only requirements for streaming table sources
  - Can Not perform streaming queries against the table

Presentation

- 4.1.2 - Books Pipeline

Poll

- A data engineer is designing a Delta Live Tables pipeline. The source system generates files containing changes captured in the source data. Each change event has metadata indicating whether the specified record was inserted, updated, or deleted. In addition to a timestamp column indicating the order in which the changes happened. The data engineer needs to update a target table based on these changes events.

  Which of the following commands can the data engineer use to best solve this problem?

A. UPDATE
B. COPY INTO
C. INSERT INTO
D. MERGE INTO
E. APPLY CHANGES INTO

Poll

- A data engineer is designing a Delta Live Tables pipeline. The source system generates files containing changes captured in the source data. Each change event has metadata indicating whether the specified record was inserted, updated, or deleted. In addition to a timestamp column indicating the order in which the changes happened. The data engineer needs to update a target table based on these changes events.

  Which of the following commands can the data engineer use to best solve this problem?

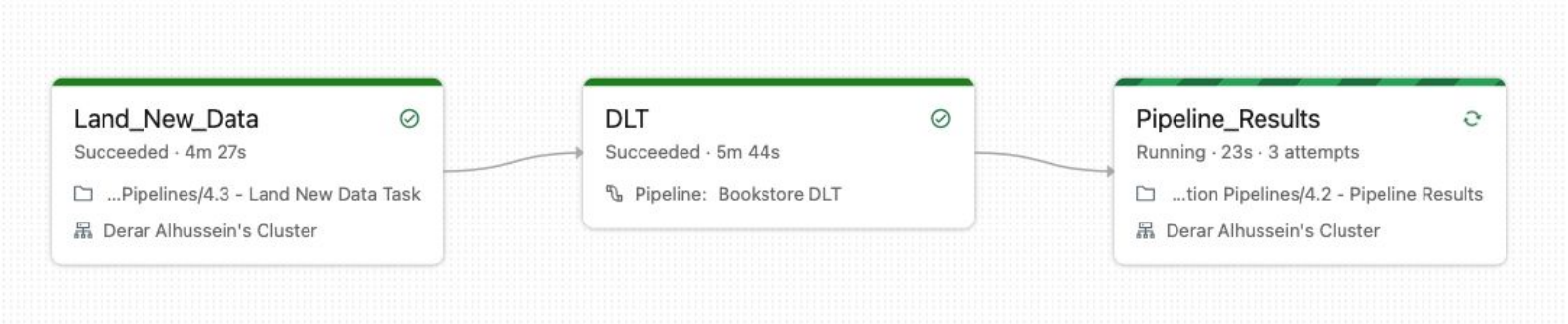
A. UPDATE
B. COPY INTO
C. INSERT INTO
D. MERGE INTO
E. APPLY CHANGES INTO

Q&A

Databricks Jobs

Databricks Jobs

- Orchestrate data pipelines by scheduling one or multiple tasks as part of a job.



Presentation

- 4.3 - Land New Data Task

Hands-on Lab

In your Databricks workspace:

- Solve Lab: **4.2L - Jobs - Land New Data**

Estimated Time: 30 Minutes

- Create your Databricks workspace in your own cloud account

Q&A

Databricks SQL

Presentation

- Databricks SQL demo

Hands-on Lab

In your Databricks workspace:

- Solve Lab: **4.3L - Databricks SQL**

Estimated Time: 30 Minutes

- Create your Databricks workspace in your own cloud account

Q&A

Data object privileges

Learning Objectives

- Data governance model of Databricks Hive metastore
- Managing Permissions for Data objects

Data governance model

- Programmatically grant, deny, and revoke access to data objects

GRANT Privilege **ON** Object <object-name> **TO** <user or group>

- **GRANT** **SELECT** **ON** **TABLE** my_table **TO** user_1@company.com

Data objects

GRANT Privilege **ON** Object <object-name> **TO** <user or group>

Object	Scope
CATALOG	controls access to the entire data catalog.
SCHEMA	controls access to a database.
TABLE	controls access to a managed or external table.
VIEW	controls access to SQL views.
FUNCTION	controls access to a named function.
ANY FILE	controls access to the underlying filesystem.

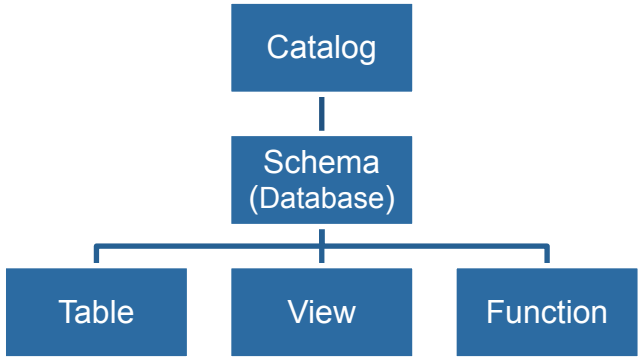
Privileges

GRANT Privilege **ON** Object <object-name> **TO** <user or group>

Privilege	Ability
SELECT	read access to an object.
MODIFY	add, delete, and modify data to or from an object.
CREATE	create an object
READ_METADATA	view an object and its metadata.
USAGE	No effect! required to perform any action on a database object.
ALL PRIVILEGES	gives all privileges

Granting Privileges by Role

Role	Can grant access privileges for
Databricks administrator	All objects in the catalog and the underlying filesystem.
Catalog owner	All objects in the catalog.
Database owner	All objects in the database.
Table owner	Only the table
…	…



More operations

- Grant
- DENY
- REVOKE

- SHOW GRANTS

Presentation

- Managing Permissions demo

Poll

- A data engineer uses the following SQL query:

  GRANT MODIFY ON TABLE employees TO hr_team

  Which of the following describes the ability given by the MODIFY privilege ?

  A. It gives the ability to add data from the table
  B. It gives the ability to delete data from the table
  C. It gives the ability to modify data in the table
  D. All the above abilities are given by the MODIFY privilege
  E. None of these options correctly describe the ability given by the MODIFY privilege

Poll

- A data engineer uses the following SQL query:

  GRANT MODIFY ON TABLE employees TO hr_team

  Which of the following describes the ability given by the MODIFY privilege ?

  A. It gives the ability to add data from the table
  B. It gives the ability to delete data from the table
  C. It gives the ability to modify data in the table
  D. All the above abilities are given by the MODIFY privilege
  E. None of these options correctly describe the ability given by the MODIFY privilege

Q&A

Unity Catalog

Learning Objectives

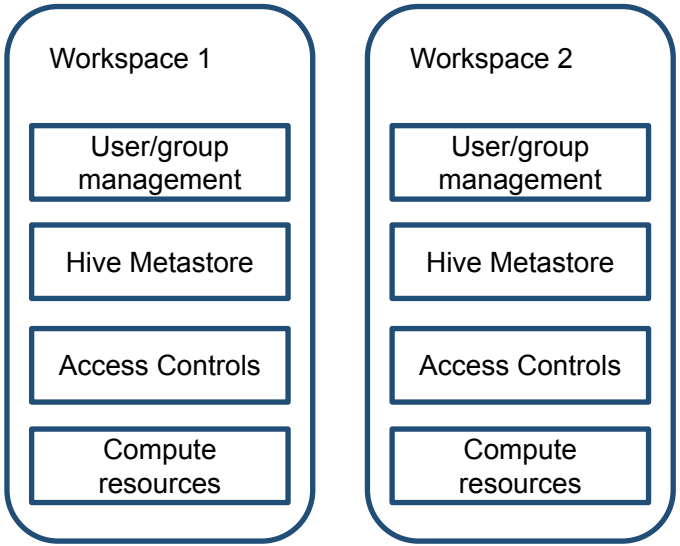
- What is Unity Catalog
- 3-level namespace
- Security model for governing data objects

Unity Catalog

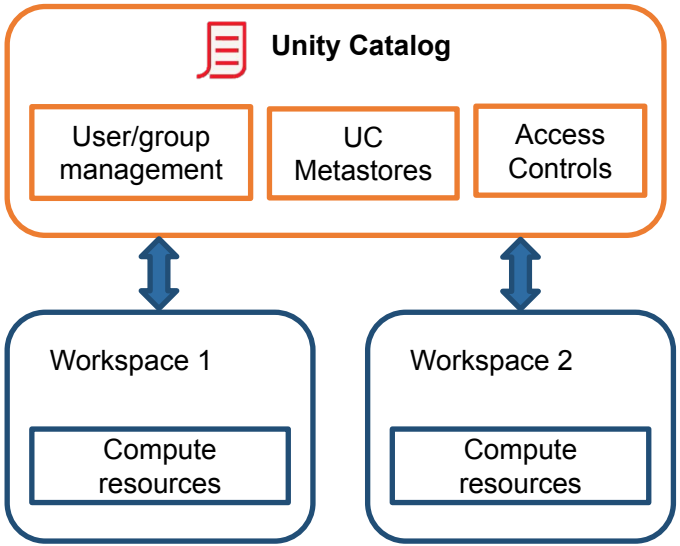
- Centralized governance solution across all your workspaces on any cloud.
- Unify governance for all data and AI assets
  - files, tables, machine learning models and dashboards
  - based on SQL

Architecture

Before Unity Catalog



With Unity Catalog

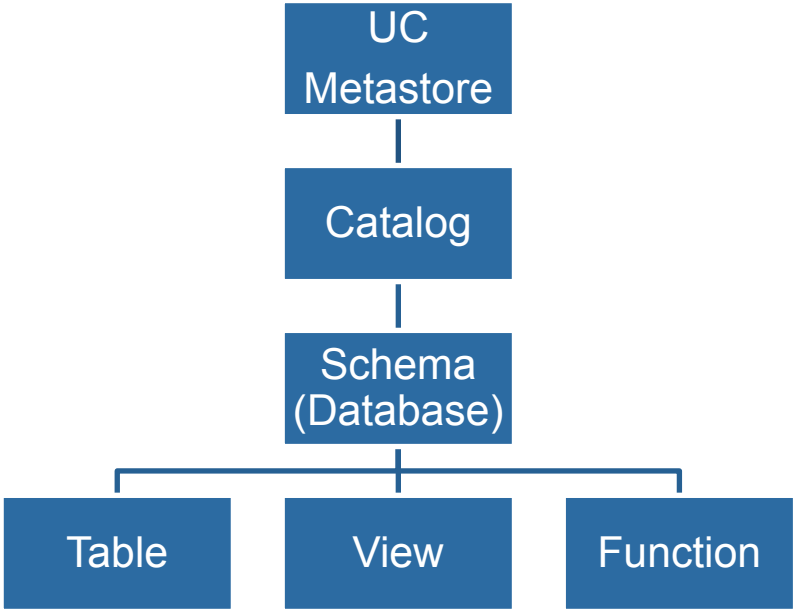


UC 3-level namespace

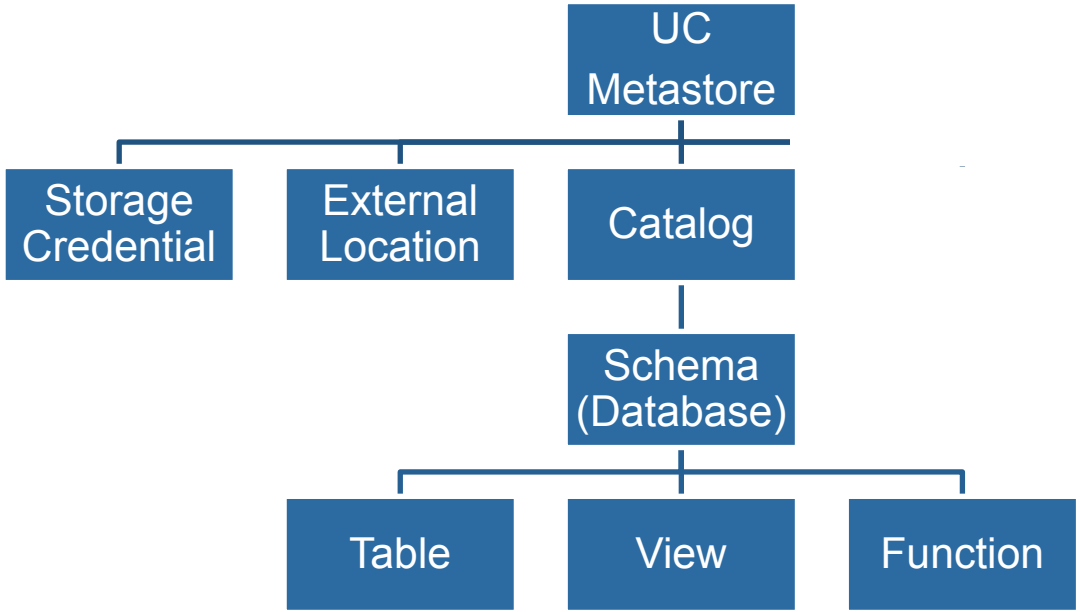
SELECT * FROM schema.table

SELECT * FROM **catalog**.schema.table

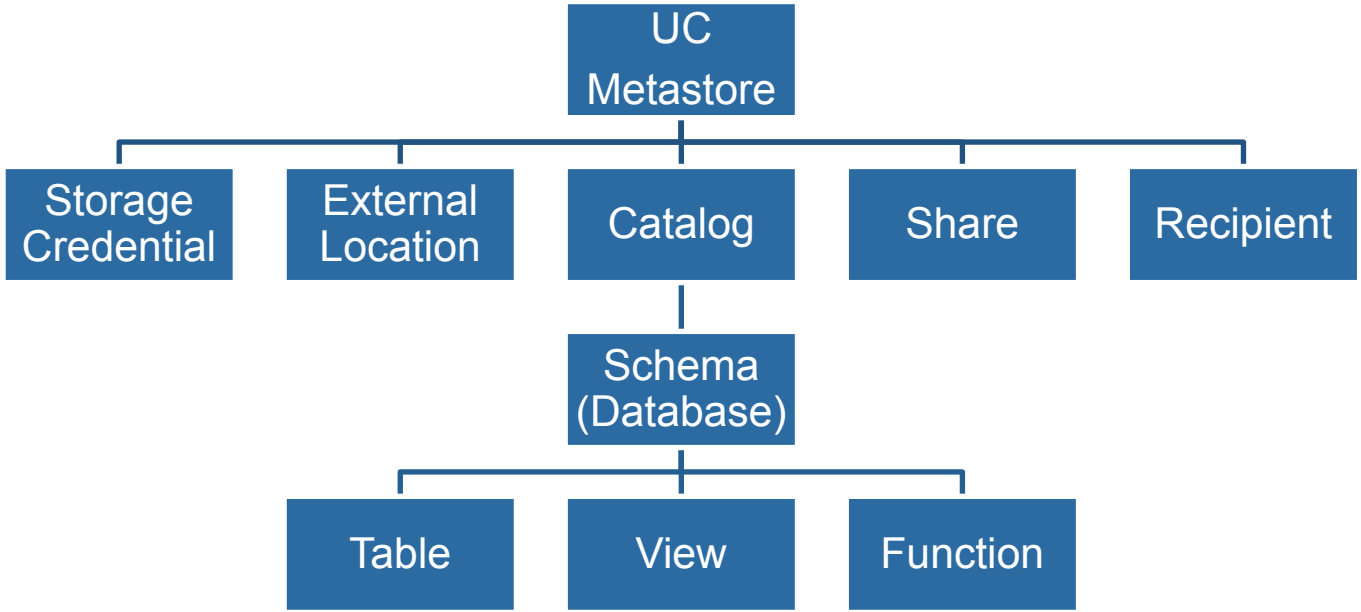
UC hierarchy



UC hierarchy



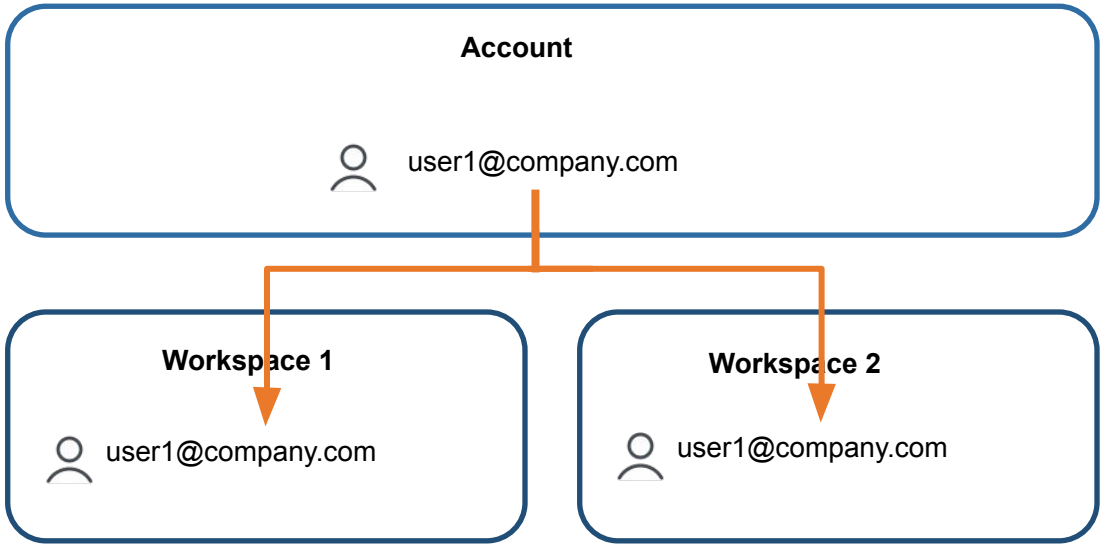
UC hierarchy



Identities

- **Users**: identified by e-mail addresses
  - Account administrator
- **Service Principles**: identified by Application IDs
  - Service Principles with administrative privilege
- **Groups**: grouping Users and Service Principles
  - Nested groups

Identity Federation

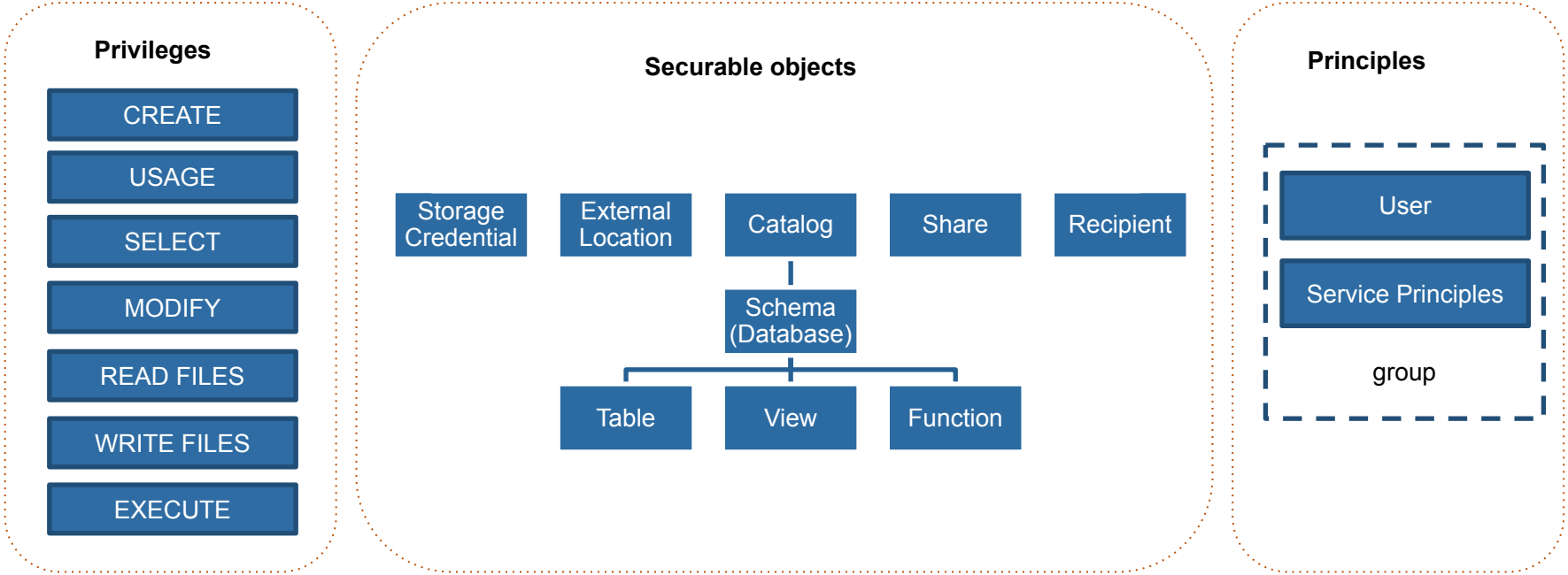


Privileges

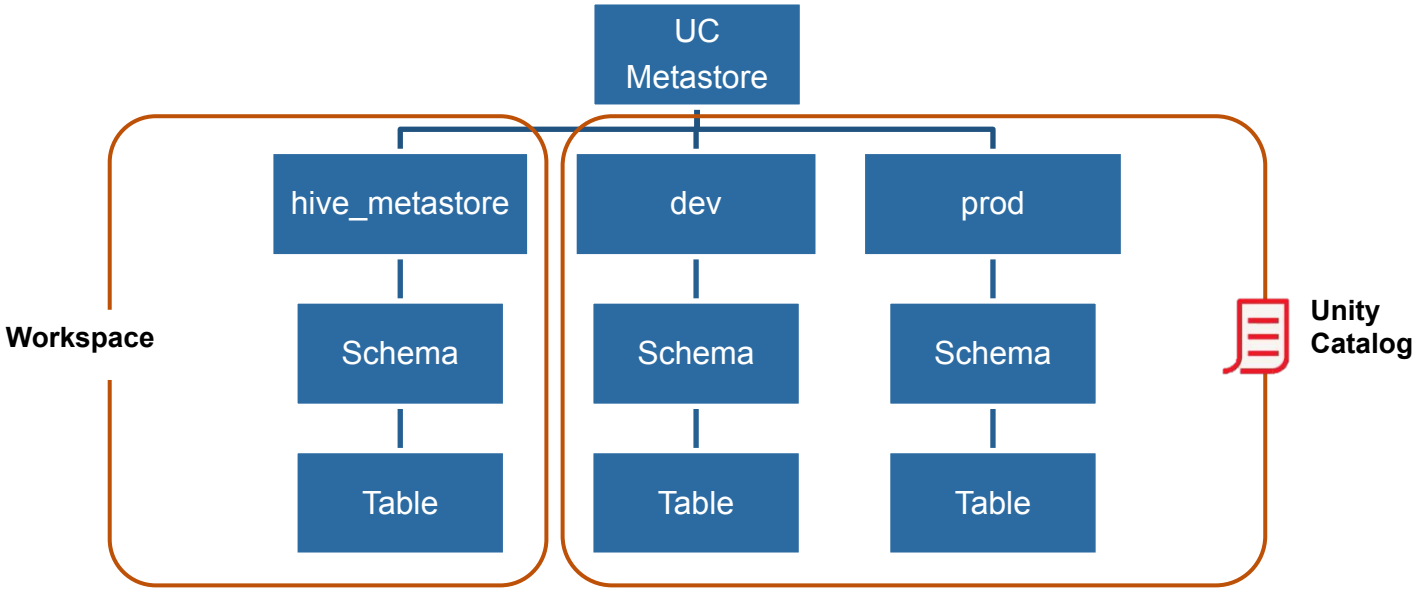
- CREATE
- USAGE
- SELECT
- MODIFY
- READ FILES
- WRITE FILES
- EXECUTE

Security model

GRANT **Privilege** ON **Securable_Object** TO **Principal**



Accessing legacy Hive metastore



Features

- Centralized governance for data and AI
- Built-in data search and discovery
- Automated lineage
- No hard migration required

Account Console

- Log in as account administrator
  - AWS: https://accounts.cloud.databricks.com
  - Azure: https://accounts.azuredatabricks.net/
  - GCP: https://accounts.gcp.databricks.com/

Presentation

- Unity Catalog demo

Poll

- Which of the following represents the hierarchy of relational entities in Unity Catalog ?

A. Metastore → Catalog → Table → Schema (Database)

B. Schema (Database) → Metastore → Catalog → Table

C. Metastore → Catalog → Schema (Database) → Table

D. Catalog → Metastore → Schema (Database) → Table

E. Schema (Database) → Catalog → Table → Metastore

Poll

- Which of the following represents the hierarchy of relational entities in Unity Catalog ?

A. Metastore → Catalog → Table → Schema (Database)

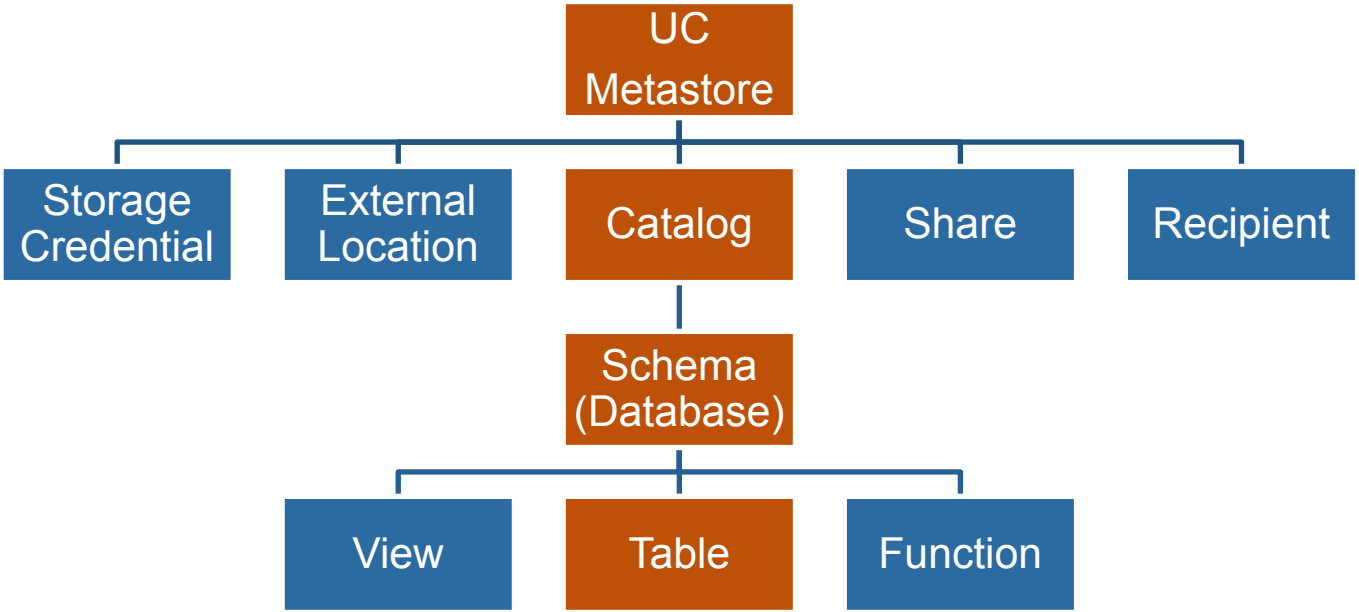
B. Schema (Database) → Metastore → Catalog → Table

C. Metastore → Catalog → Schema (Database) → Table

D. Catalog → Metastore → Schema (Database) → Table

E. Schema (Database) → Catalog → Table → Metastore

UC hierarchy



Q&A

Certification Overview

Learning Objectives

- Format and structure of the exam.
- The topics covered in the exam.
- Types of questions provided on the exam

Exam Details

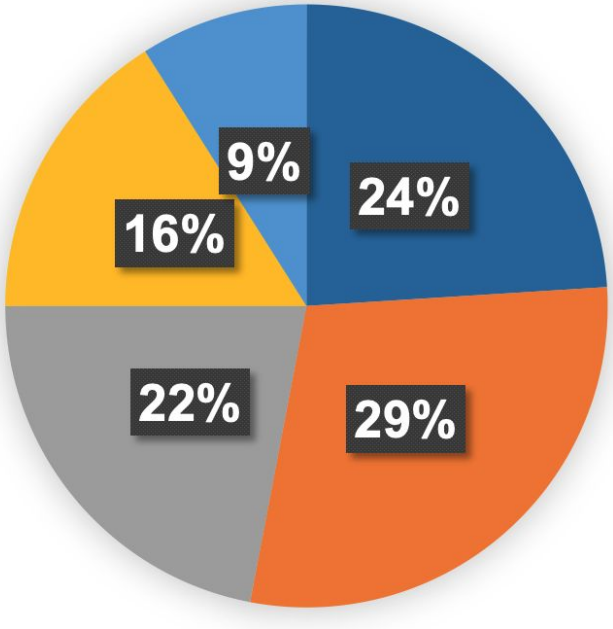
- Time allotted to complete exam = 1.5 hours

- Number of Questions = 45
- Passing scores = At least 70% (32/45)

- Exam fee = $200
- Exam retake fee = $200

- Exam Guide: https://www.databricks.com/learn/certification/data-engineer-associate

Exam Questions



- Databricks Lakehouse Platform (11/45)
- ELT with Spark SQL and Python (13/45)
- Incremental Data Processing (10/45)
- Production Pipelines (7/45)
- Data Governance (4/45)

Out-of-scope

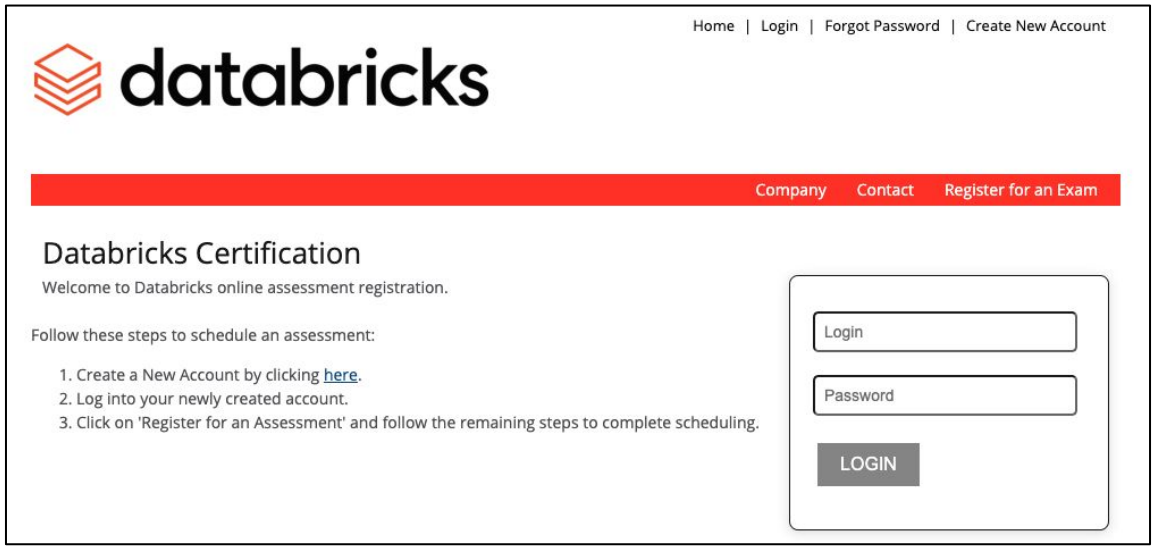
- Apache Spark internals
- Databricks CLI and REST API
- Change Data Capture CDC/CDF
- Data modeling concepts
- GDPR/CCPA
- Monitoring and logging production jobs
- Dependency management
- Testing

Code Examples

- Code will be in mainly in SQL
- Otherwise, Python

Exam Platform

https://www.webassessor.com/databricks



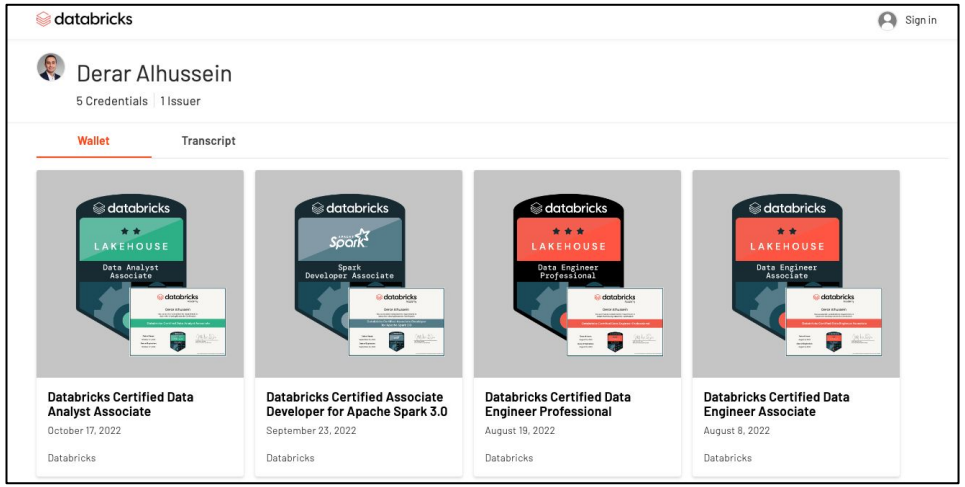
Exam Proctoring

- Monitored via webcam by a Webassessor proctor.
- Asked to provide valid, photo-based identification.
- Proctor does not provide assistance on the content of the exam
- No test aids will be available during the exam

Exam Result

- Certification exams are automatically graded.
- 24 hours to receive the badge and Certificate

https://credentials.databricks.com



Questions Types

- Multiple-choice questions – Only one correct answer
- Questions Types:
  - Conceptual Questions
  - Code-Based Questions

Conceptual Questions

- Which part of the Databricks Lakehouse Platform can data engineers use to orchestrate jobs ?

A. Repos
B. Workflows
C. Data Explorer
D. Databricks SQL
E. Cluster

Conceptual Questions

- Which part of the Databricks Lakehouse Platform can data engineers use to orchestrate jobs ?

A. Repos
B. Workflows
C. Data Explorer
D. Databricks SQL
E. Cluster

Code-Based Questions

```
spark.table("sales")
      .writeStream
      .option("checkpointLocation", checkpointPath)
      .______
      .table("new_sales")
```

If you want the query to execute a single micro-batch to process all of the available data, which of the following lines of code should you use to fill in the blank ?

A. trigger(once=True)
B. trigger(continuous="once")
C. processingTime("once")
D. trigger(processingTime="once")
E. processingTime(1)

Code-Based Questions

```
spark.table("sales")
      .writeStream
      .option("checkpointLocation", checkpointPath)
      .______
      .table("new_sales")
```

If you want the query to execute a single micro-batch to process all of the available data, which of the following lines of code should you use to fill in the blank ?

A. trigger(once=True)
B. trigger(continuous="once")
C. processingTime("once")
D. trigger(processingTime="once")
E. processingTime(1)

Practice Exam

- Databricks official Practice test
- PDF file: https://files.training.databricks.com/assessments/practice-exams/PracticeExam-DataEngineerAssociate.pdf

Q&A

Derar Alhussein

www.derar.cloud

www.linkedin.com/in/DerarAlhussein


O'REILLY®