Lakehouse Architecture Basics

Day 3 - Databricks 14-Day AI Challenge

Yash Kavaiya

Gen AI Guru | Easy AI Labs

January 11, 2026

Agenda

- **Data Architectures Evolution** - From Warehouses to Lakehouse
- **What is a Data Lakehouse?** - Combining the best of both worlds
- **Delta Lake** - The foundation of Lakehouse
- **Lakehouse Components** - Complete architecture overview
- **ACID Transactions** - Ensuring data reliability
- **Medallion Architecture** - Bronze, Silver, Gold layers

The Journey to Modern Data Architecture

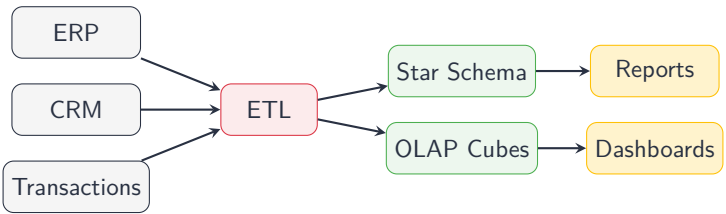
Phase 1: Data Warehouses (1980s-2000s)

- Structured data only
- ACID transactions
- SQL-based querying
- High cost per TB

Phase 2: Data Lakes (2010s)

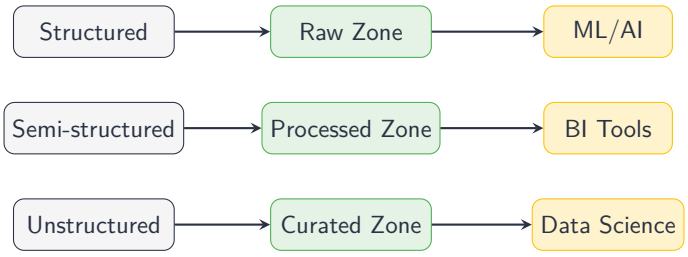
- All data types
- Schema-on-read
- Low cost storage
- Poor reliability

Phase 1: Data Warehouses



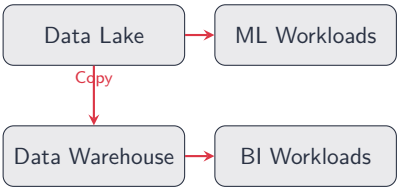
Limitation: Not suitable for unstructured data (images, logs, JSON)

Phase 2: Data Lakes



Problem: No ACID transactions → **"Data Swamps"**

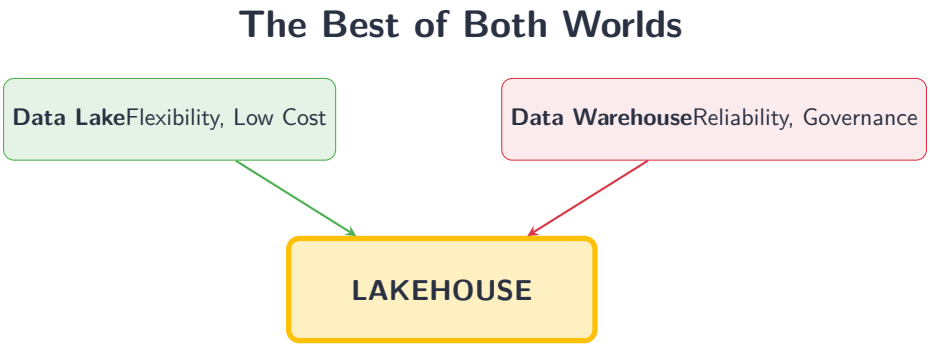
The Two-Tier Architecture Problem



Problems:

- • Data duplication
- • Stale data
- • Complex pipelines
- • High costs
- • Governance challenges

What is a Data Lakehouse?



Key Lakehouse Principles

Principle	Description	Benefit
Open Formats	Data in Parquet, Delta	No vendor lock-in
ACID Transactions	Reliable updates	Data consistency
Schema Enforcement	Data quality at write	Prevent bad data
Time Travel	Access previous versions	Auditing, rollback
Unified Access	Same data for all	No data silos
Direct Access	Query in place	No copying needed

Delta Lake: The Foundation

What is Delta Lake?

An **open-source storage layer** that brings reliability to Data Lakes.

How it works:

- Stores data as **Parquet files**
- Adds a **transaction log**
- Enables ACID on object storage

Table Structure:

```
delta_table/
+-- _delta_log/
|   +-- 000...00.json
|   +-- 000...01.json
|   +-- 000...02.json
+-- part-00000.parquet
+-- part-00001.parquet
+-- part-00002.parquet
```

Delta Lake Features

Feature	Description	Example
ACID Transactions	Atomic, reliable operations	Multiple concurrent writers
Time Travel	Query historical data	`VERSION AS OF 5`
Schema Evolution	Add columns safely	`ALTER TABLE ADD COL`
Schema Enforcement	Reject invalid data	Wrong types fail
Audit History	Track all changes	`DESCRIBE HISTORY`
MERGE (Upserts)	Update/Insert combined	CDC processing

Time Travel in Action

Query Any Version of Your Data

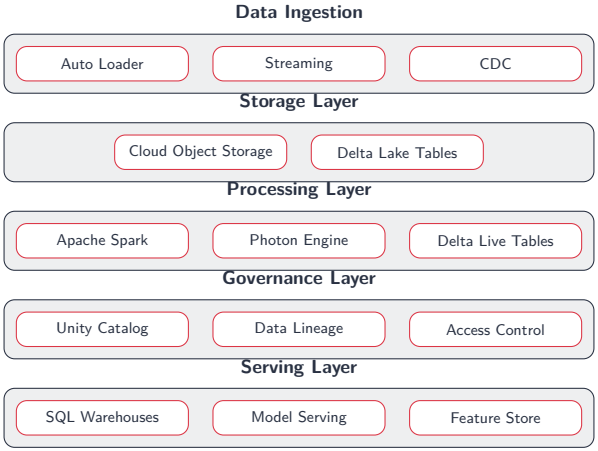
Query by Version:

```
-- Current version
SELECT * FROM sales;

-- Specific version
SELECT * FROM sales
VERSION AS OF 5;
```

Query by Timestamp:

```
-- Point in time
SELECT * FROM sales
TIMESTAMP AS OF
'2024-01-01';

-- Restore data
RESTORE TABLE sales
TO VERSION AS OF 5;
```

Complete Lakehouse Architecture



Data Ingestion Layer

Auto Loader

- Incremental file ingestion
- Schema inference
- Exactly-once semantics

Streaming

- Kafka integration
- Event Hubs
- Kinesis support

CDC

- Database changes
- Real-time sync
- MERGE operations

Processing & Governance Layers

Processing Layer

- **Apache Spark:** Core engine
- **Photon Engine:** C++ vectorized (2-8x faster)
- **Delta Live Tables:** Declarative ETL

Governance Layer

- **Unity Catalog:** Centralized governance
- **Data Lineage:** Track data flow
- **Access Control:** Row/column security

Ensuring Data Reliability

Property	Definition	Example
Atomicity	All succeed or all fail	Debit AND credit both happen
Consistency	Data always valid	Balance never negative
Isolation	No interference	Two concurrent updates
Durability	Survives failures	Power outage safe

How Delta Lake Achieves ACID

1. Optimistic Concurrency

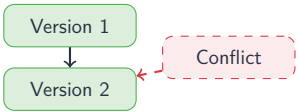
- Writers don't block
- Conflicts at commit
- Automatic retry

2. Write-Ahead Log

- Log before data
- Enables recovery

3. Atomic Commits

- New log entry = complete
- Incomplete writes ignored



Medallion Architecture

Bronze → Silver → Gold

Bronze Layer - Raw Data

Characteristics:

- Raw data “as-is” from source
- No transformations
- Append-only writes
- Full history preserved
- Schema-on-read

Example:

- Raw JSON from IoT sensors
- CSV files from vendors
- API response dumps

Quality: Low - may have errors, duplicates

Silver Layer - Cleansed Data

Characteristics:

- Cleaned and validated
- Deduplication applied
- Data types enforced
- Joined with reference data
- Schema enforced

Example:

- Parsed sensor readings
- Typed and validated
- Deduplicated records

Quality: Medium - business rules applied

Gold Layer - Business Ready

Characteristics:

- Business-level aggregations
- Denormalized for performance
- Optimized for use cases
- Multiple tables per domain

Example:

- Hourly averages by location
- Feature tables for ML
- KPI dashboards

Quality: High - analytics ready

Key Takeaways

- **Lakehouse** = Data Lake flexibility + Data Warehouse reliability
- **Delta Lake** provides ACID transactions on object storage
- **Time Travel** enables auditing and rollback capabilities
- **Medallion Architecture:** Bronze → Silver → Gold
- **Unity Catalog** provides centralized governance

One platform for all data workloads!

Thank You!

Questions?

Connect with me:

linkedin.com/in/yashkavaiya

Gen AI Guru