Using SQL to Query your Data Lake with Delta Lake

Easily Build your Data Lakehouse with Apache Spark 3.0 and Delta Lake

Denny Lee dennyglee

August 2020

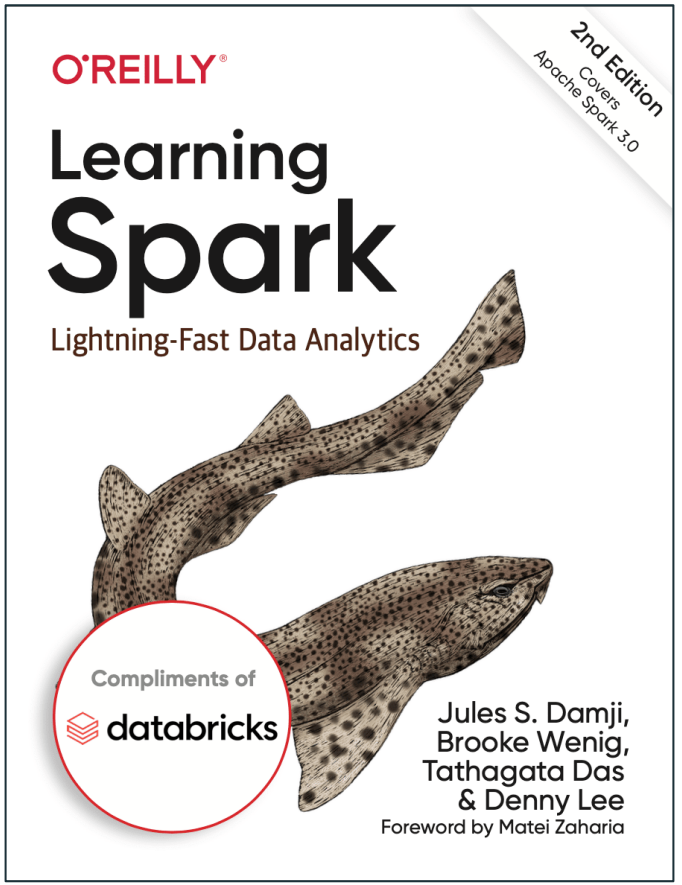
databricks

About the Speaker

Denny Lee is a Developer Advocate at Databricks. He is a hands-on distributed systems and data sciences engineer with extensive experience developing internet-scale infrastructure, data platforms, and predictive analytics systems for both on-premise and cloud environments.

O'Reilly Learning Spark 2nd Edition for free

http://dbricks.co/get-ebook



Go to delta.io to Read the VLDB Paper!

Delta Lake: High-Performance ACID Table Storage over Cloud Object Stores

Michael Armbrust, Tathagata Das, Liwen Sun, Burak Yavuz, Shixiong Zhu, Mukul Murthy, Joseph Torres, Herman van Hovell, Adrian Ionescu, Alicja Łuszczak, Michał Świtakowski, Michał Szafrański, Xiao Li, Takuya Ueshin, Mostafa Mokhtar, Peter Boncz[1], Ali Ghodsi[2], Sameer Paranjpye, Pieter Senster, Reynold Xin, Matei Zaharia[3]
Databricks, [1]CWI, [2]UC Berkeley, [3]Stanford University
delta-paper-authors@databricks.com

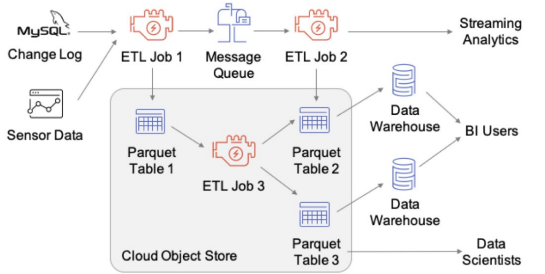
ABSTRACT

Cloud object stores such as Amazon S3 are some of the largest and most cost-effective storage systems on the planet, making them an attractive target to store large data warehouses and data lakes. Unfortunately, their implementation as key-value stores makes it difficult to achieve ACID transactions and high performance: metadata operations such as listing objects are expensive, and consistency guarantees are limited. In this paper, we present Delta Lake, an open source ACID table storage layer over cloud object stores initially developed at Databricks. Delta Lake uses a transaction log that is compacted into Apache Parquet format to provide ACID properties, time travel, and significantly faster metadata operations for large tabular datasets (e.g., the ability to quickly search billions of table partitions for those relevant to a query). It also leverages this design to provide high-level features such as automatic data layout optimization, upserts, caching, and audit logs. Delta Lake tables can be accessed from Apache Spark, Hive, Presto, Redshift and other systems. Delta Lake is deployed at thousands of Databricks customers that process exabytes of data per day, with the largest instances managing exabyte-scale datasets and billions of objects.

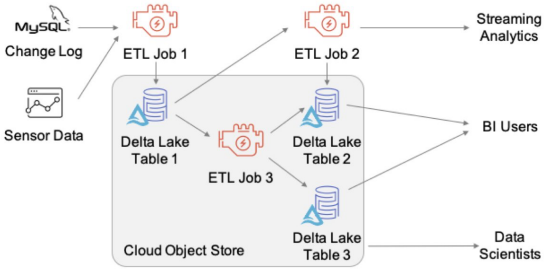
The major open source "big data" systems, including Apache Spark, Hive and Presto [45, 52, 42], support reading and writing to cloud object stores using file formats such as Apache Parquet and ORC [13, 12]. Commercial services including AWS Athena, Google BigQuery and Redshift Spectrum [1, 29, 39] can also query directly against these systems and these open file formats.

Unfortunately, although many systems support reading and writing to cloud object stores, achieving *performant* and *mutable* table storage over these systems is challenging, making it difficult to implement data warehousing capabilities over them. Unlike distributed filesystems such as HDFS [5], or custom storage engines in a DBMS, most cloud object stores are merely key-value stores, with no cross-key consistency guarantees. Their performance characteristics also differ greatly from distributed filesystems and require special care.

The most common way to store relational datasets in cloud object stores is using columnar file formats such as Parquet and ORC, where each table is stored as a set of objects (Parquet or ORC "files"), possibly clustered into "partitions" by some fields (e.g., a separate set of objects for each date) [45]. This approach can offer acceptable performance for scan workloads as long as the object



(a) Pipeline using separate storage systems.



(b) Using Delta Lake for both stream and table storage.

databricks

databricks

Unified data analytics platform for accelerating innovation across data science, data engineering, and business analytics

Global company with 5,000 customers and 450+ partners

Original creators of popular data and machine learning open source projects

Webinar

How Apache Spark 3.0 and Delta Lake Enhance Data Lake Reliability

Delta Lake 0.7.0 is the first release on Apache Spark 3.0 and adds support for metastore-defined tables and SQL DDLs

Support for defining tables in the Hive metastore

With Spark 3.0 metastore support, you can define Delta tables in the Hive metastore and use the table name in all SQL operations

```
-- Create Table
CREATE TABLE events (
 date DATE,
 eventId STRING,
 eventType STRING,
)
```

```
-- Alter Table Schema
ALTER TABLE events
  ADD COLUMNS (
    eventCategory STRING
 )
```

Support for SQL Delete, Update and Merge

SQL - not just for inserts any more!

```
-- Create Table
CREATE TABLE events (
 date DATE,
 eventId STRING,
 eventType STRING,
)
```

```
-- Delete events
DELETE FROM events WHERE
date < '2017-01-01'
```

```
-- Alter Table Schema
ALTER TABLE events
  ADD COLUMNS (
    eventCategory STRING
  )
```

```
-- Update events
UPDATE events SET
eventType = 'click' WHERE
eventType = 'click'
```

```
-- Insert into table
INSERT INTO events
SELECT * FROM newEvents
```

```
-- Upsert data to a target Delta
-- table using merge
MERGE INTO events
USING updates
ON events.eventId = updates.eventId
WHEN MATCHED THEN UPDATE
     SET events.data = updates.data
WHEN NOT MATCHED THEN INSERT
     (date, eventId, data)
     VALUES (date, eventId, data)
```

But what happens with DMLs under the covers?

What really happens to the file system when you run delete, update, and merge?

```
DELETE FROM events ...
UPDATE events ...
MERGE INTO events ...
```

But what happens with DML under the covers?

What really happens to the file system when you run delete, update, and merge?

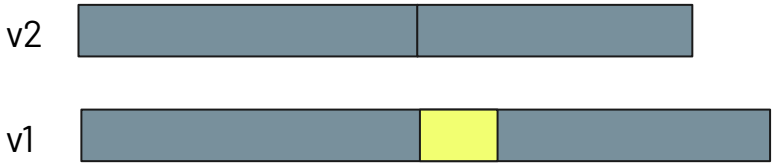
```
DELETE FROM events ...
UPDATE events ...
MERGE INTO events ...
```

v1

But what happens with DML under the covers?

What really happens to the file system when you run delete, update, and merge?

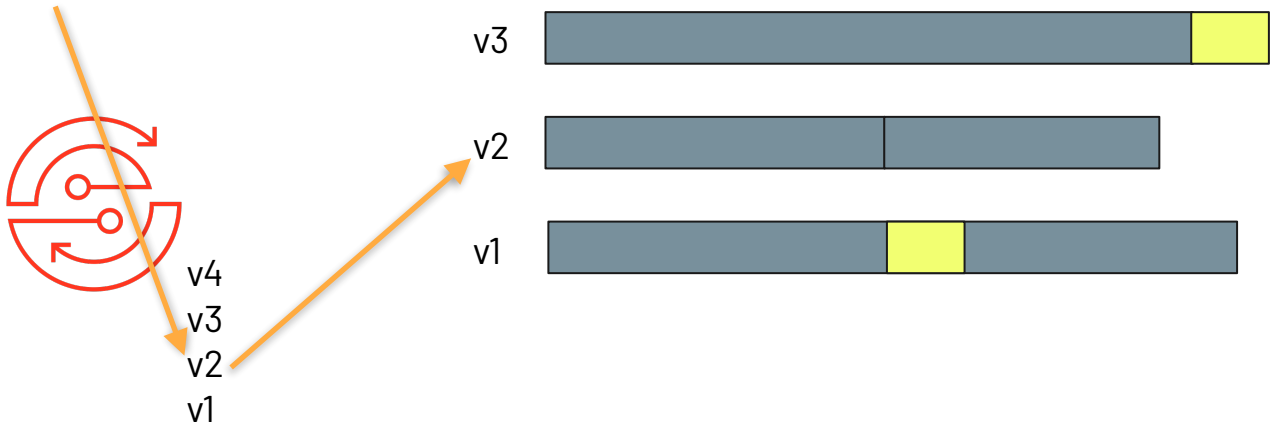
```
DELETE FROM events ...
UPDATE events ...
MERGE INTO events ...
```



Time Travel

The transaction log and additive files = data versioning

```
SELECT * FROM events
VERSION AS OF 2
```



Control Table History Retention

- `delta.logRetentionDuration`: controls how long the history for a table is kept.
- `delta.deletedFileRetentionDuration`: controls how long ago a file must have been deleted before being a candidate for VACUUM.

```
ALTER TABLE delta.`pathToDeltaTable`
SET TBLPROPERTIES(
   delta.logRetentionDuration = "interval <interval>"
   delta.deletedFileRetentionDuration = "interval <interval>"
)
```

Enable DataSourceV2 and Catalog API Integration

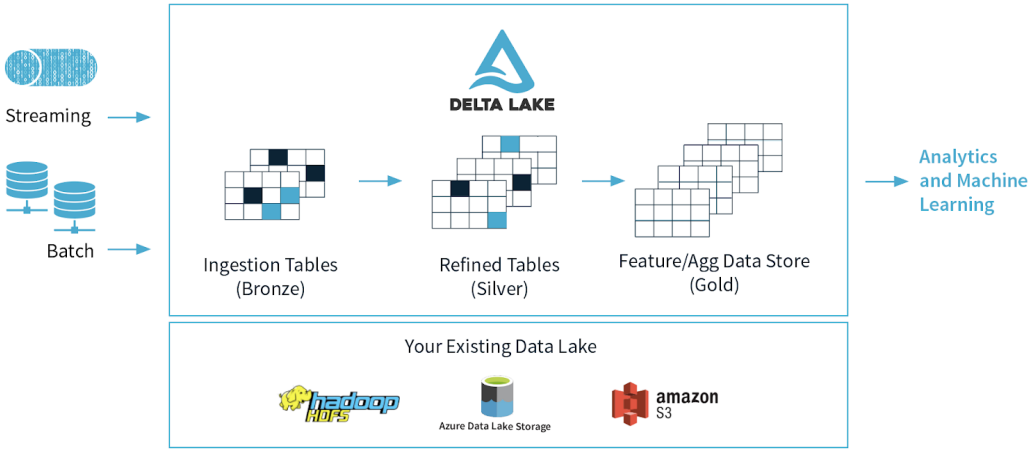
To use the aforementioned features, you must enable the integration with Apache Spark DataSourceV2 and Catalog APIs (since 3.0) by setting the following configurations when creating a new SparkSession.

```
from pyspark.sql import SparkSession

spark = SparkSession \
  .builder \
  .appName("...") \
  .master("...") \
  .config("spark.sql.extensions", "io.delta.sql.DeltaSparkSessionExtension") \
  .config(“spark.sql.catalog.spark_catalog", "org.apache.spark.sql.delta.catalog.DeltaCatalog") \
  .getOrCreate()
```

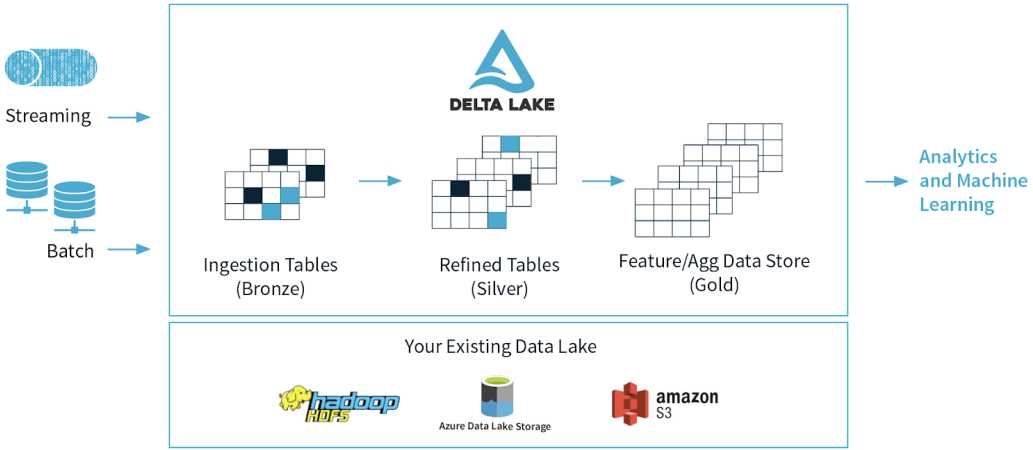
Data Quality Framework

Improved SQL DDL and DMLs and ACID Transactions are just the start



Lakehouse Paradigm

Improved Performance, DW-like capabilities, on low cost cloud object stores



Adaptive Query Execution

Dynamic Partition Pruning

Query Compilation Speedup

Join Hints

Data Source V2 API + Catalog Support

DDL/DML Enhancements

DELETE/UPDATE/ MERGE in Catalyst

Demo Time

databricks

Build your own Delta Lake
at **https://delta.io**

Try out Spark 3.0 + Delta Lake now!

- Try out Spark 3.0 and Delta Lake now using Databricks Community Edition at databricks.com/try
- Try out the notebooks available at https://github.com/databricks/tech-talks
- Learn more by joining us at the Data + AI Online meetup: https://www.meetup.com/data-ai-online/
- Get the free O'Reilly Learning Spark 2nd Edition eBook at http://dbricks.co/get-ebook (incl Spark 3.0 and Delta Lake)
- Read the VLDB paper: Delta Lake: High-Performance ACID Table Storage over Cloud Object Stores at delta.io