Enforcing and Evolving the Schema

Diving into Delta Lake Series

Andreas Neumann + Denny Lee

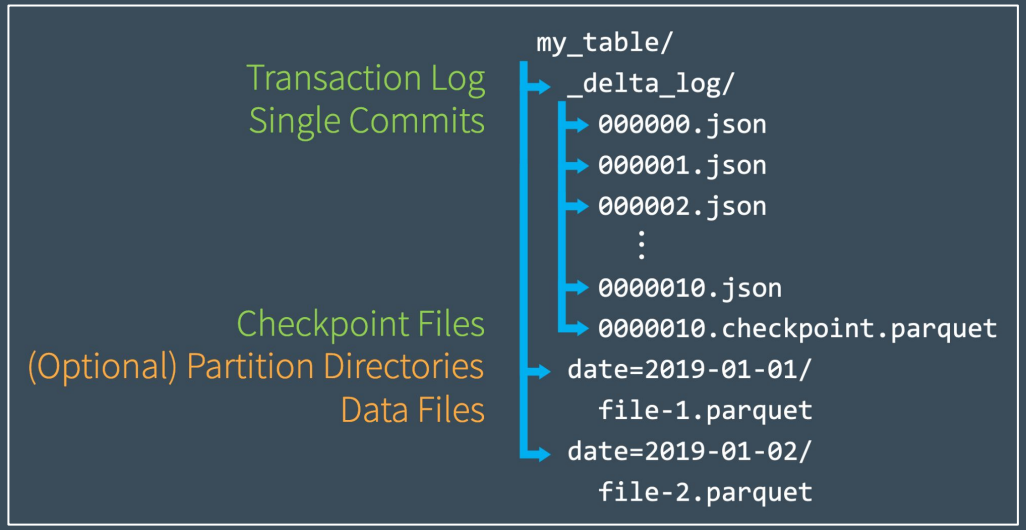
Who are we?

databricks

- Staff Software Engineer – Databricks
- Dedicated to Delta and Data Pipelines
- Building Pipelines on Apache Spark since 2014
- Formerly Created Big Data Systems at Scale in Google Cloud, Cask Data, Yahoo!, IBM
- C.S. Master from TU Dortmund, Germany
- C.S. PhD from Uni Trier, Germany

Who are we?

- Staff Developer Advocate – Databricks
- Working with Apache Spark™ since v0.6
- Former Senior Director Data Science Engineering at Concur
- Former Microsoftie: Cosmos DB, HDInsight (Isotope)
- Masters Biomedical Informatics - OHSU
- BS in Physiology - McGill



In the previous session, we discussed
Unpacking the Transaction Log

Overview

- Data, like our experiences, is always evolving and accumulating
- As business problems and requirements evolve over time, so too does the structure of your data
- With Delta Lake, as the data changes, incorporating new dimensions is easy
  - schema enforcement: prevents users from accidentally polluting their tables with mistakes or garbage data
  - schema evolution: enables automatic addition of columns when desired

Understanding Table Schemas

- Apache Spark™ DataFrames contain the schema
- With Delta Lake, the table's schema is saved in JSON format inside the transaction log.

```
schemaString: {"type":"struct","fields":[
  {"name":"loan_id","type":"long","nullable":false,"metadata":{}},
  {"name":"funded_amnt","type":"integer","nullable":true,"metadata":{}},
  {"name":"paid_amnt","type":"double","nullable":true,"metadata":{}},
  {"name":"addr_state","type":"string","nullable":true,"metadata":{}}
]}
```

What is Schema Enforcement?

- Schema enforcement, also known as *schema validation*, rejects writes to a table that does not match the table's schema
- Schema validation occurs *on write*
- If the schema is not compatible, Delta Lake *cancels* the transaction, i.e. no data is written
- As well, Delta Lake raises an exception to let the user know about the mismatch.

Schema Enforcement Rules

- **Cannot contain any additional columns** that are not present in the target table's schema.
- It's OK if the incoming data doesn't contain every column in the table – those columns will simply be assigned null values.
- Will fail if those columns are not nullable.

Schema Enforcement Rules

- **Cannot have column data types that differ from the column data types in the target table**.
- E.g., target table's column contains `StringType` data, but corresponding source column contains `IntegerType` data, schema enforcement will raise an exception and prevent the write operation from taking place.

Schema Enforcement Rules

- **Can not contain column names that differ only by case**.
- e.g. cannot have columns such as '`Foo`' and '`foo`' defined in the same table.
- Notes:
  - Spark can be used in case sensitive or insensitive (default) mode,
  - Parquet is case sensitive when storing and returning column information.
  - Delta Lake is case-preserving but insensitive when storing the schema.
  - This restriction has been added to avoid potential mistakes, data corruption or loss issues

How Is Schema Enforcement Useful?

Use schema enforcement as a gatekeeper of a clean, fully transformed data production. It's typically enforced on tables that directly feed:

- Machine learning algorithms
- BI dashboards
- Data analytics and visualization tools
- Any production system requiring highly structured, strongly typed, semantic schemas

In order to prepare their data for this final hurdle, many users employ a simple "multi-hop" architecture that progressively adds structure to their tables.

What is Schema Evolution?

- Schema evolution allows users to easily change a table's current schema to accommodate data that is changing over time.
- Most commonly used operations for
  - append
  - overwrite
- Use `.option('mergeSchema', 'true')` to your `.write` or `.writeStream` Spark command.
- Also can use `spark.databricks.delta.schema.autoMerge = True` to Spark configuration.
- Use with caution, as schema enforcement will no longer warn you about unintended schema mismatches.

What is Schema Evolution?

- With `.option('mergeSchema', 'true')`
- “Read-compatible” schema changes
- During table appends or overwrites
- The following types of schema changes are eligible
  - Adding new columns (this is the most common scenario)
  - Changing of data types from `non-nullable` to `nullable`,
  - Upcasts from `ByteType -> ShortType -> IntegerType`

What is Schema Evolution?

- With `.option("overwriteSchema", "true")`
- Non-”read-compatible” schema changes
- Typically when overwriting
- The following types of schema changes:
  - Dropping a column
  - Changing an existing column’s data type (in place)
  - Renaming column names that differ only by case (e.g. “Foo” and “foo”)

- Spark 3.0 will include DDL using `ALTER TABLE`

Demo

Delta Lake Connectors

Standardize your big data storage with an open format accessible from various tools

Delta Lake Partners and Providers

More and more partners and providers are working with Delta Lake

Google Dataproc

Privacera

Informatica

Streamsets

WANDisco

Qlik

Users of Delta Lake

Tencent 腾讯 · Alibaba Group · COMCAST · viacom · ciena · JAM CITY

edmunds · databricks · healthdirect Australia · Booz | Allen | Hamilton · McAfee Together is power. · CONDÉ NAST

TILTING POINT · Mc Graw Hill Education · acorns · EVERQUOTE · Namely · usermind

ZEISS · split · upwork · DOLLAR SHAVE CLUB · wehkamp

How do I use  DELTA LAKE ?

Get Started with Delta using Spark APIs

Add Spark Package

```
pyspark --packages io.delta:delta-core_2.12:0.5.0

bin/spark-shell --packages io.delta:delta-core_2.12:0.5.0
```

Maven

```
<dependency>
  <groupId>io.delta</groupId>
  <artifactId>delta-core_2.12</artifactId>
  <version>0.5.0</version>
</dependency>
```

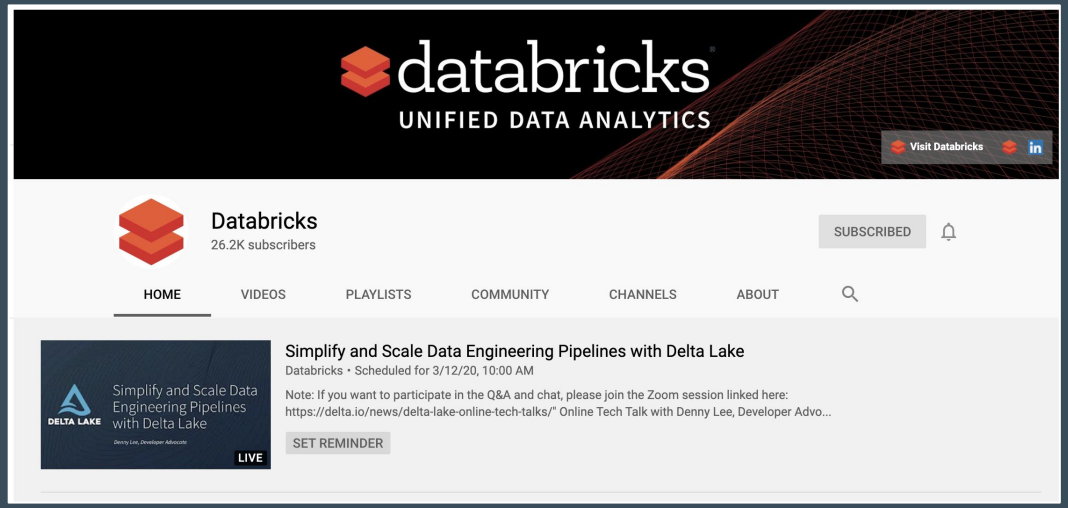
Instead of `parquet`...

```
dataframe
    .write
    .format("parquet")
    .save("/data")
```

... simply say `delta`

```
dataframe
    .write
    .format("delta")
    .save("/data")
```

Subscribe Today!



https://dbricks.co/youtube

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