databricks

Databricks Delta:

Bringing Unprecedented Reliability and Performance to Cloud Data Lakes

AN “UNDER THE HOOD” LOOK

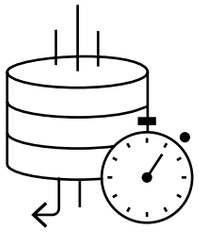
Databricks Delta, a component of the Databricks Unified Analytics Platform*, is a unified data management system that brings unprecedented reliability and performance (10-100 times faster than Apache Spark on Parquet) to cloud data lakes. Designed for both batch and stream processing, it also addresses concerns regarding system complexity. Its advanced architecture enables high reliability and low latency through the use of techniques such as schema validation, compaction, data skipping, etc. to address pipeline development, data management and as well query serving.

* Databricks Unified Analytics Platform, from the original creators of Apache Spark™,accelerates innovation by unifying data science, engineering and business.

Content

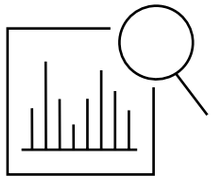
Challenges in Harnessing Data

Many organizations have responded to the their ever-growing data volumes by adopting data lakes as places to collect their data ahead of making it available for analysis. While this has tended to improve the situation somewhat data lakes suffer from some key challenges of their own:



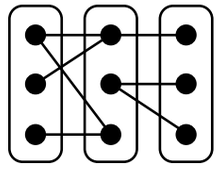
QUERY PERFORMANCE

The required ETL processes can add significant latency such that it may take hours before incoming data manifests in a query response so the users do not benefit from the latest data. Further, increasing scale and the resulting longer query run times can prove unacceptably long for users.



DATA RELIABILITY

The complex data pipelines are error-prone and complex consuming inordinate resources. Further, schema evolution as business needs change can be effort-intensive. Finally, errors or gaps in incoming data, a not uncommon occurrence, can cause failures in downstream applications.



SYSTEM COMPLEXITY

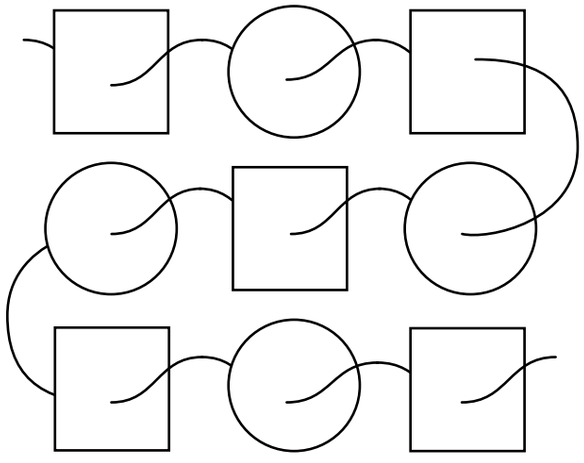
It is difficult to build flexible data engineering pipelines that combine streaming and batch analytics. Building such systems requires complex and low-level code. Interventions during stream processing with batch correction or programming multiple streams from the same sources or to the same destinations is restricted.

Challenges in Harnessing Data

Practitioners typically organize their pipelines using a multi-hop architecture. The pipeline starts with a “firehose” of records from many different parts of the organization. These data are then normalized and enriched with dimension information. Following this it may be filtered down and aggregated for particular business objectives. Finally, high-level summaries of key business metrics might be created.

There are various challenges encountered through the pipeline stages:

- Schema changes can break enrichment, joins, transforms between stages
- Failures may cause data between stages to either drop on the floor or be duplicated
- Partitioning alone does not scale for multi-dimensional data
- Standard tables do not allow combining streaming and batch for best latencies
- Concurrent access suffer from inconsistent query results
- Failing streaming jobs can require resetting and restarting data processing

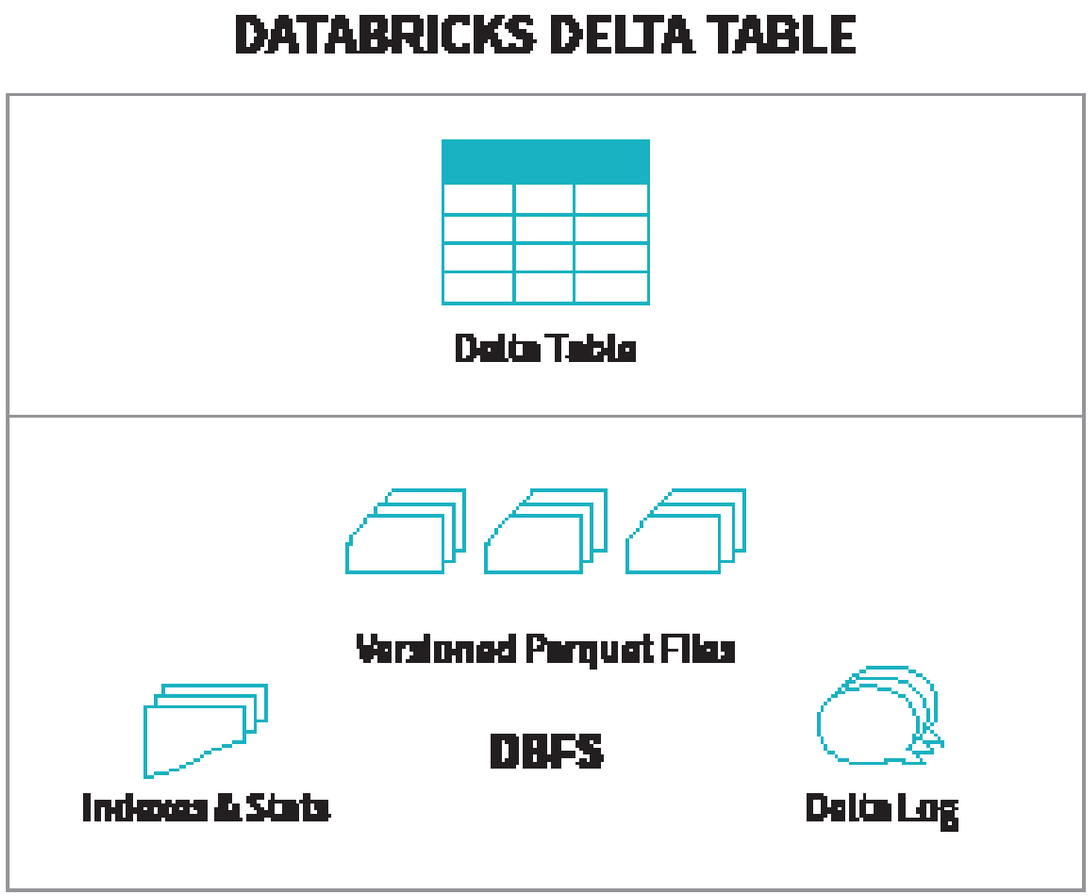


Databricks Delta addresses the challenges faced by data engineering professionals in marshalling their data head-on by providing the opportunity for a much simpler analytics architecture able to address both batch and stream use case with high query performance and high data reliability.

Databricks Delta Architecture

The Delta architecture provides an efficient and transactional way to work with large data sets stored as files on S3 (or other blob stores). It employs an ordered log (the Delta Log) of atomic collections of actions (e.g. AddFile, RemoveFile, etc.). It is based on the notion of Databricks Delta tables built atop the Databricks File System (DBFS) which manifests:

- Versioned Parquet files (based on Apache Parquet[1])
- Indexes and stats
- The Delta log



1 https://parquet.apache.org/

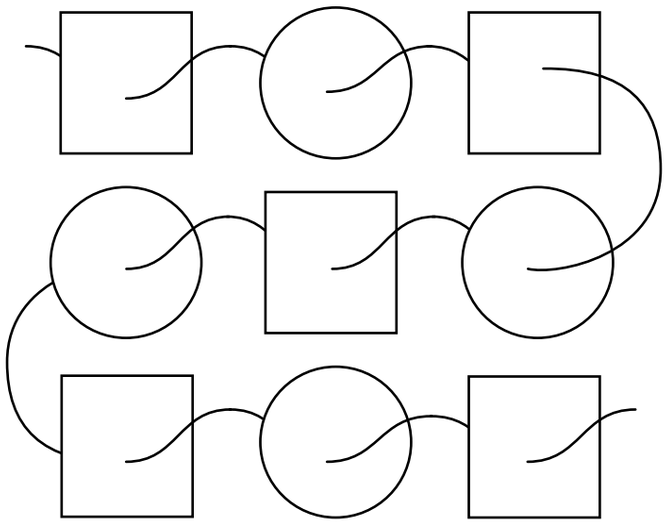
Databricks Delta Architecture

Parquet is a columnar storage format and supports very efficient compression and encoding schemes. In Delta the versioned Parquet files enable tracking the evolution of the data. Indexes and statistics about the files are maintained to increase query efficiency. The Delta log can be appended to by multiple writers that are mediated using optimistic concurrency control providing serializable ACID transactions. Changes to the table are stored as ordered atomic units called commits. The log is designed such that it can be read in parallel by a cluster of Spark executors.

The Delta design allows readers to efficiently query a snapshot of the state of a table, optionally filtering by partition value such that you get fast operations irrespective of the number of files.

Building and Maintaining Robust Pipelines

Designing and building robust pipelines is the first step in realizing value from one's data resources. The focus needs to be not only on ingesting the data but also on ensuring quality upon ingest and being able to maintain the data over time as enhancements and corrections need to be made and manage the environment effectively as queries as being run. Delta helps address challenges throughout the various pipeline stages while helping you address both batch and streaming data.

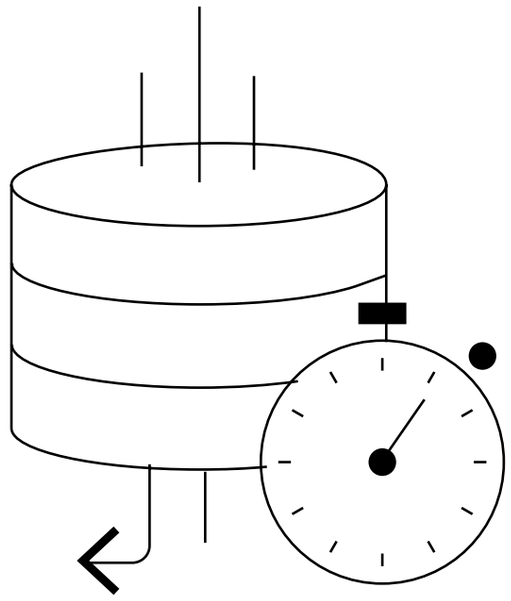


Delta Details: Query Performance

Delta uses a number of techniques to address query performance, data reliability and system complexity:

Query Performance

Query performance is a major driver of user satisfaction. The faster a query returns results, the sooner those results are available for using. With Delta we have observed query performance 10 to 100 times faster than with Apache Spark on Parquet. Delta employs various techniques to deliver superior performance.



Delta Details: Query Performance

DATA INDEXING

Delta creates and maintains indexes of the ingested data. This speeds up the querying dramatically.

DATA SKIPPING

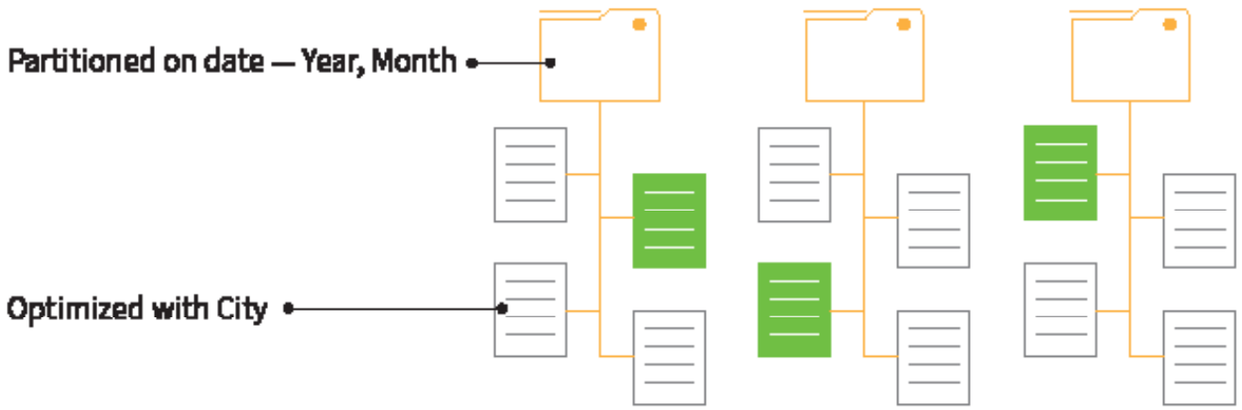
Delta maintains file statistics so that data subsets relevant to the query are used instead of entire tables — this partition pruning avoids processing data that is not relevant to the query. Multi-dimensional clustering (using Z-ordering algorithm) is used to enable this. This techniques is particularly helpful in the case of complex queries.

COMPACTION

Often, especially in the case of streaming data, a large number of small files are created as data is ingested. Storing and accessing these small files can be processing-intensive, slow and inefficient from a storage utilization perspective. Delta manages file sizes (i.e. compacts or combines multiple small files into more efficient larger ones) to speed up query performance.

DATA CACHING

Accessing data from storage repeatedly can slow query performance. Delta automatically caches highly accessed data to speed access for queries improving performance by an order of magnitude.

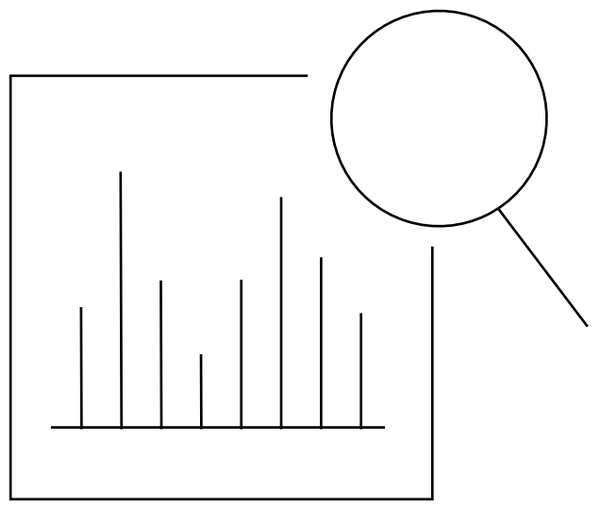


Only read the relevant subsets (shown in green) to fulfil query

Delta Details: Data Reliability

Data Reliability

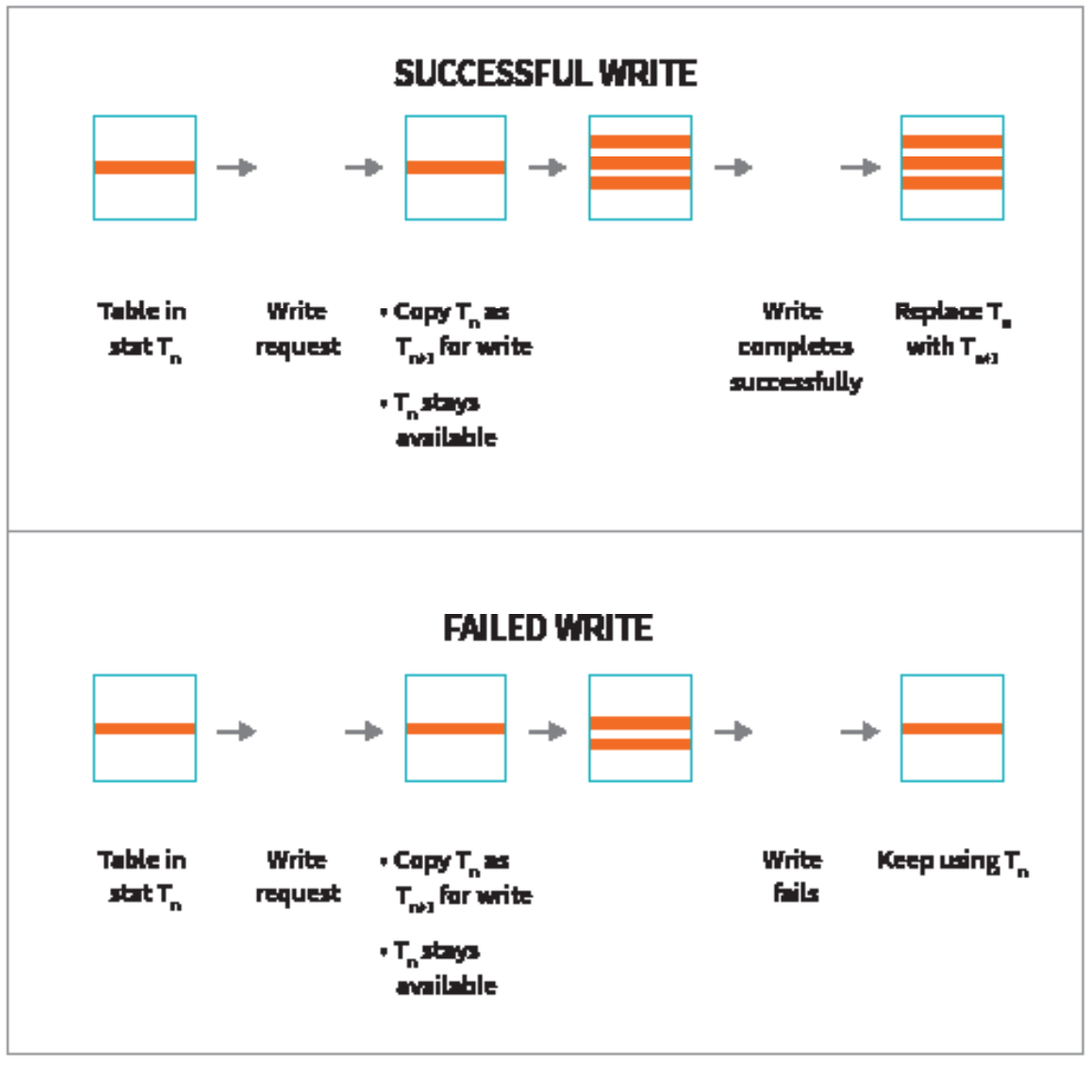
Reliable datasets are key to successful data analytics and data use whether that be feeding dashboards or enabling ML initiatives. Delta uses various techniques to achieve data reliability.



Delta Details: Data Reliability

ACID TRANSACTIONS

The inevitable partial or failed writes risk corrupting the data. Delta employs an “all or nothing” ACID transaction approach to prevent such corruption.



SNAPSHOT ISOLATION

In large environments with multiple concurrent readers and writers it is critical that metadata be maintained in a way such that reads in progress act on consistent views of data and are not impacted by writes in progress. Delta provides snapshot isolation ensuring that multiple writers can write to a dataset simultaneously without interfering with jobs reading the dataset.

SCHEMA ENFORCEMENT

Notionally similar data but from different sources or of different vintage can differ in its representation creating difficulties for using it effectively. Delta helps ensure data integrity for ingested data by providing schema enforcement so that data can be stored using the preferred schema and avoids potential data corruption with incorrect or invalid schemas.

EXACTLY ONCE

When working with long running computations, multiple streams or with concurrent batch jobs there is a risk that some data is missed (due to transmission difficulties) or duplicated (in attempts to correct for the misses). Delta employs checkpointing to provide a robust exactly once delivery semantic to ensure that data are neither missed nor repeated erroneously.

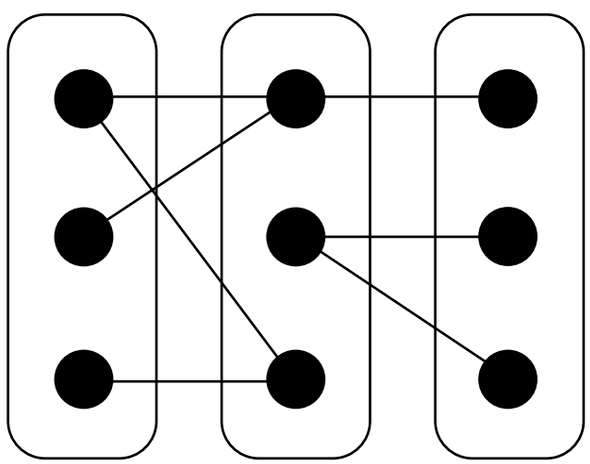
UPSERTS AND DELETES SUPPORT

Standard Spark tables are write-once i.e. they cannot be modified. Any necessary changes to take account of late-arriving data or data requiring updating must be addressed using new tables. Delta provides support for UPSERTS and DELETES making it easier to address these situation i.e. these commands provide a more “convenient” way of dealing with such changes.

Delta Details: System Complexity

System Complexity

System complexity is a key determinant not only of reliability and cost-effectiveness but very importantly, also of responsiveness. As business requirements evolve the data analytics architecture needs to be flexible and responsive to keep up.



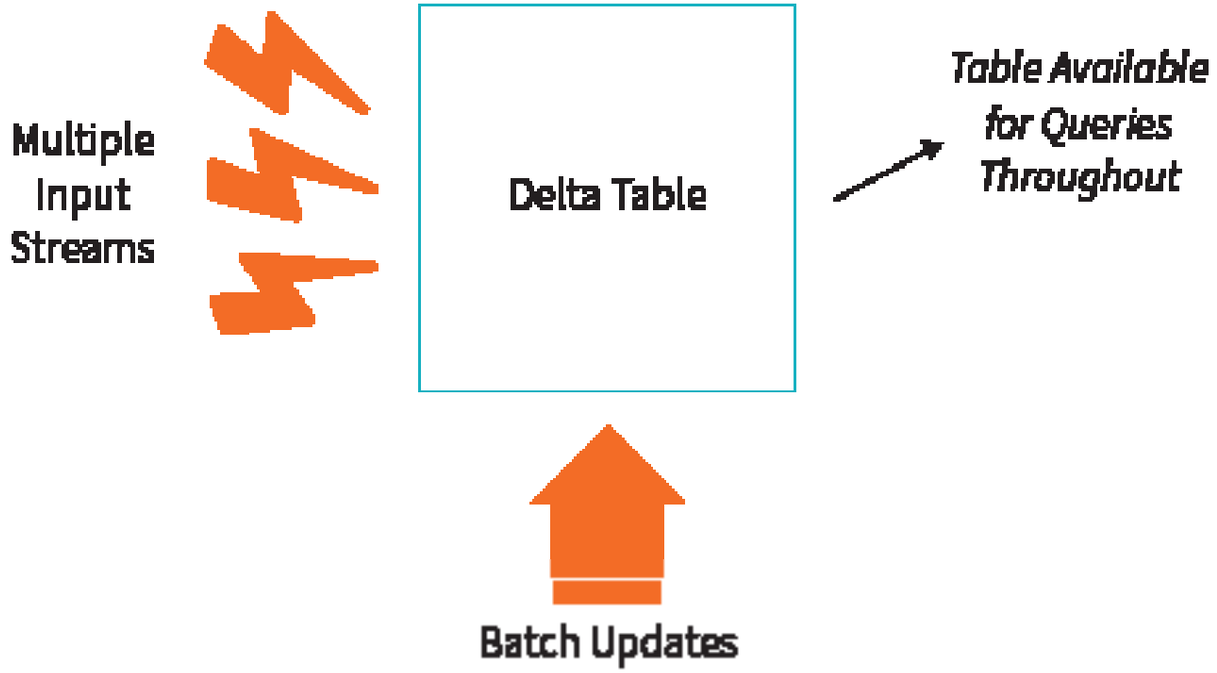
Delta Details: System Complexity

UNIFIED BATCH/STREAM

Delta is able to handle both batch and streaming data (via a direct integration with Structured Streaming for low latency updates) including the ability to concurrently write batch and streaming data to the same data table. Not only does this result in a simpler system architecture, it also results in shorter time from data ingest to query result.

SCHEMA EVOLUTION

Delta provides ability to infer schema from input data. This reduces the effort for dealing with schema impact of changing business needs at multiple levels of the pipeline/ data stack.



Delta Best Practices

Delta is a powerful new approach to managing data in a cloud data lake for analysis and ML uses cases. Adopting the following best practices will help you make the most of Delta:

Go Through Delta

All writes and reads should go through Delta to ensure consistent overall behavior. Further, Delta tables **must not** be accessed using earlier (pre-Delta) versions of Databricks Runtime because those versions do not understand Delta and do not support it.

Run OPTIMIZE Regularly

The OPTIMIZE command is used to trigger compaction. It makes no data related changes to the table, so a read before and after an OPTIMIZE has the same results. It should be run regularly on tables that analysts are querying to ensure efficiency. A good starting point is to do this on a daily basis. Note that OPTIMIZE **should not** however be run on base or staging tables.

Delta Best Practices

Run VACUUM Regularly

To ensure that concurrent readers can continue reading a stale snapshot of a table, Databricks Delta leaves deleted files on DBFS for a period of time. The VACUUM command helps save on storage costs by cleaning up these invalid files. It can, however, interrupt users querying a Delta table similar to when partitions are re-written. VACUUM should be run regularly to clean up expired snapshots that are no longer required.

Batch Modifications

Parquet files, that form the underpinning of Delta, are immutable and thus need to be rewritten completely to reflect changes regardless of the extent of the change. Use MERGE INTO to batch changes to amortize costs.

Use DELETEs

Manually deleting files from the underlying storage is likely to break the Delta table so instead you should use DELETE commands to ensure proper progression of the change.

Trying Databricks Delta

Delta is easy to put into production — Databricks customers have been able to get into production with a Delta-based solution in just a few weeks using a small team compared to alternate approaches that take much longer and more resources. It is easy to get started with Delta.

Porting existing Spark code for using Delta is as simple as changing

```
“CREATE TABLE ... USING parquet” to
“CREATE TABLE ... USING delta”
```

or changing

```
“dataframe.write.format(“parquet”).load(“/data/events”)”
“dataframe.write.format(“delta”).load(“/data/events”)”
```

Trying Databricks Delta

If you are already using Azure Databricks Premium you can explore Delta today using:

- Databricks Delta quickstart notebook for a simple `"hello world"` with Databricks Delta
- OPTIMIZE notebook to try out Databricks Delta's indexing and statistics capabilities and see how Delta's OPTIMIZE and ZORDER commands accelerate queries

If you are not already using Azure Databricks Premium, you can try Delta for free by signing up for the **free trial** and using the notebooks above.

You can learn more about Delta from the **Databricks Delta documentation**.