Delta Change Data Feed

Joseph Torres
Rahul Mahadev
Itai Weiss

DATA+AI SUMMIT 2021
FORMERLY SPARK+AI SUMMIT

DELTA LAKE

Agenda

- Data capture challenges
- Change Data Feed on the Lakehouse
  - Capture Changes
  - Process Changes
- Demo

Who are we? - Jose Torres

databricks

- Software Engineer - Databricks
- Committer - Delta Lake, Apache Spark
- Database and cooking enthusiast

Who are we? - Rahul Mahadev

databricks

- Software Engineer– Databricks
- Delta Lake Committer
- MS Computer Science, University of Illinois Urbana-Champaign

Who are we? - Itai Weiss

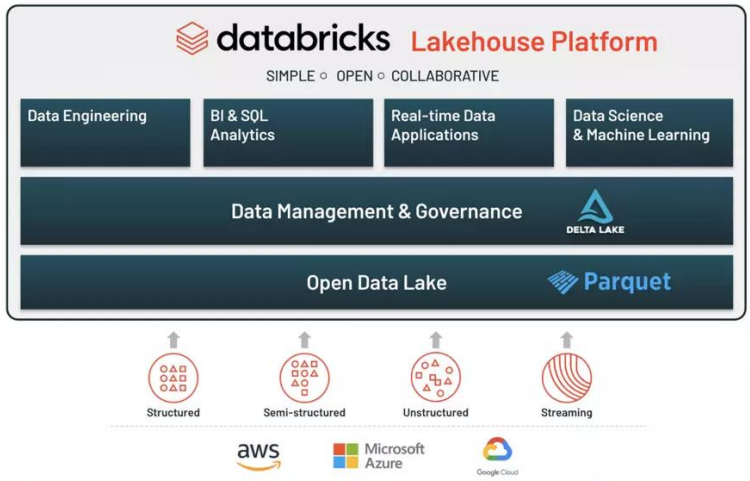
databricks

- Lead Solution Architect– Databricks
- Working with Apache Sparkˆ since v1.6
- Consulted for numerous firms across Financial, Insurance, Tech, Pharma, Manufacturing, Retail and Transportation

Delta Lake

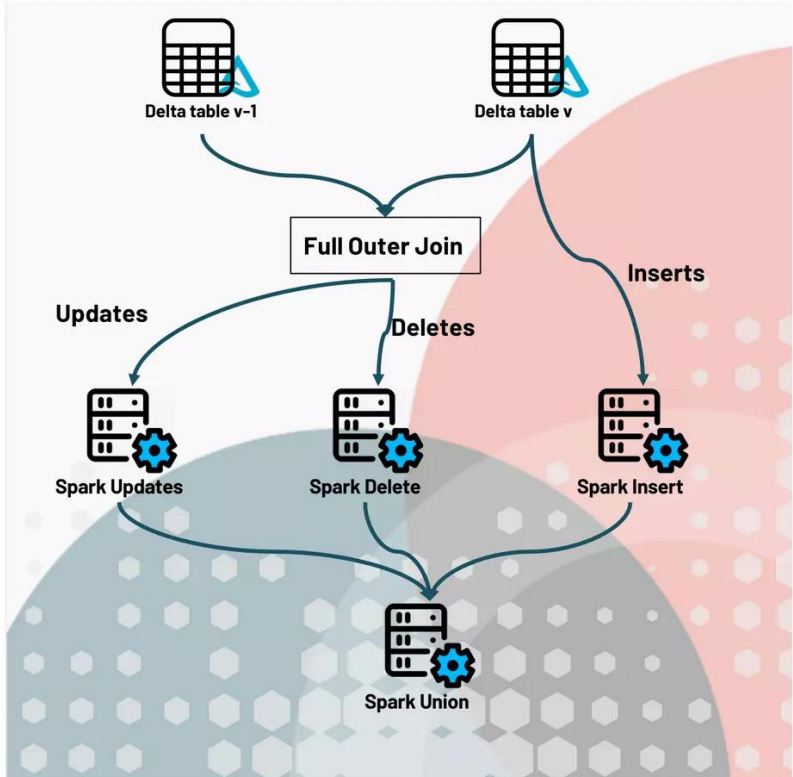
An open, reliable, performant, and secure data storage and management layer for your data

- Fresh & reliable data with a single source of truth
- Data warehouse performance with data lake economics
- Advanced security and standards to meet compliance needs
- Open format



Current Challenges

- Big data increases the complexity of
  - Lots of data
  - Changes infrequently
  - Just want to process the newest changes



Solve data challenges with CDF

- Read only the changed data
- Avoid full table scans
- Reduce compute and memory

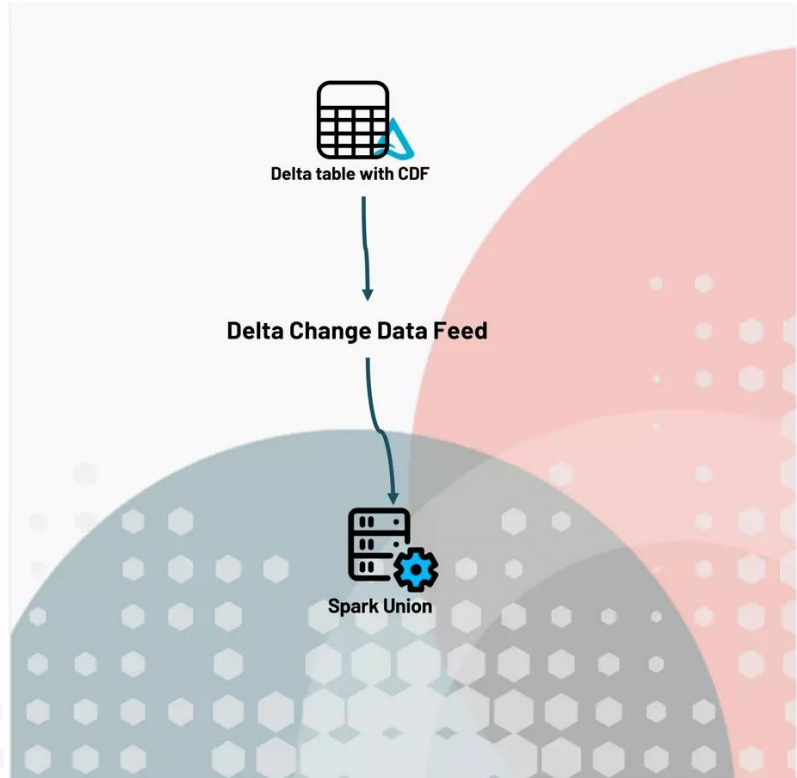
For All your use cases

Improve ETL pipelines

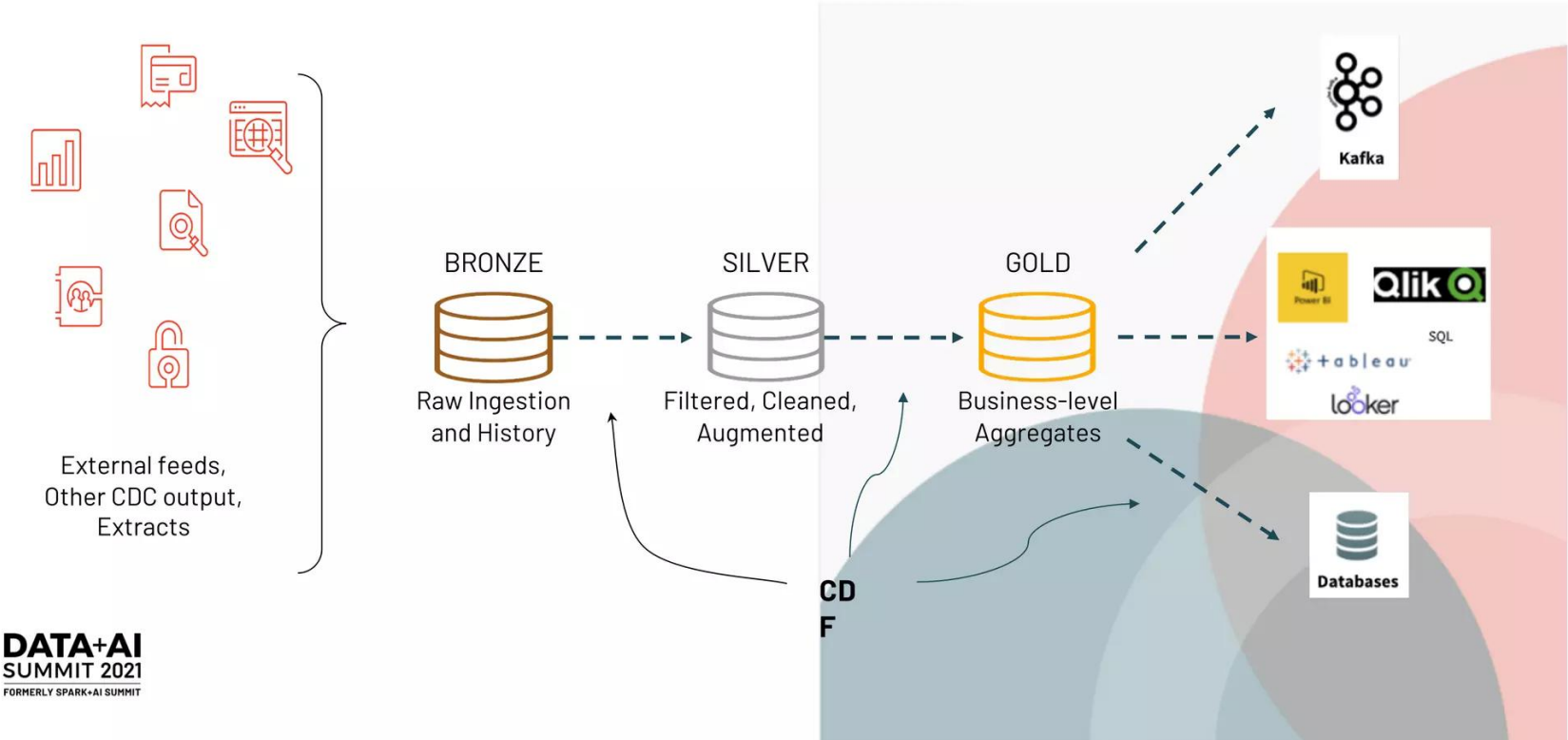
Unify batch and streaming

BI on your data lake

Meet regulatory needs



Where Delta Change Data Feed Applies



How Does Delta Change Data Feed Work?

Original Table (v1)

PK	B
A1	B1
A2	B2
A3	B3

Change data (Merged as v2)

PK	B
A2	Z2
A3	B3
A4	B4

Change Data Feed Output

PK	B	Change Type	Time	Version
A2	B2	Preimage	12:00:00	2
A2	Z2	Postimage	12:00:00	2
A3	B3	Delete	12:00:00	2
A4	B4	Insert	12:00:00	2

So it will not be output by CDF.

Consuming the Delta Change Data Feed

Stream - micro batches

12:00 12:08 12:10:10

A2	B2	Preimage	12:00:00	2
A2	Z2	Postimage	12:00:00	2
A3	B3	Delete	12:00:00	2
A4	B4	Insert	12:00:00	2

A5	B5	Insert	12:08:00	3

A6	B6	Insert	12:09:00	4
A6	B6	Preimage	12:10:05	5
A6	Z6	Postimage	12:10:05	5

12:00

A5	B5	Insert	12:08:00	3
A6	B6	Insert	12:09:00	4

12:10:00

A6	B6	Preimage	12:10:05	5
A6	Z6	Postimage	12:10:05	5

12:20

Batch - every 10 mins

Batch Consumption

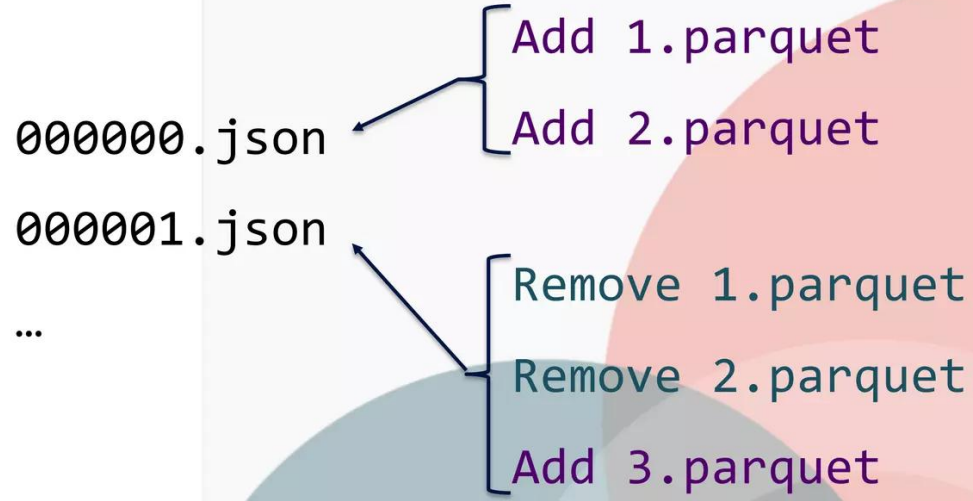
- Batches are constructed based in time-bound windows which may contain multiple Delta versions

Stream-based Consumption

- Delta Change Feed is processed as each source commit completes
- Rapid source commits can result in multiple Delta versions being included in a single micro-batch

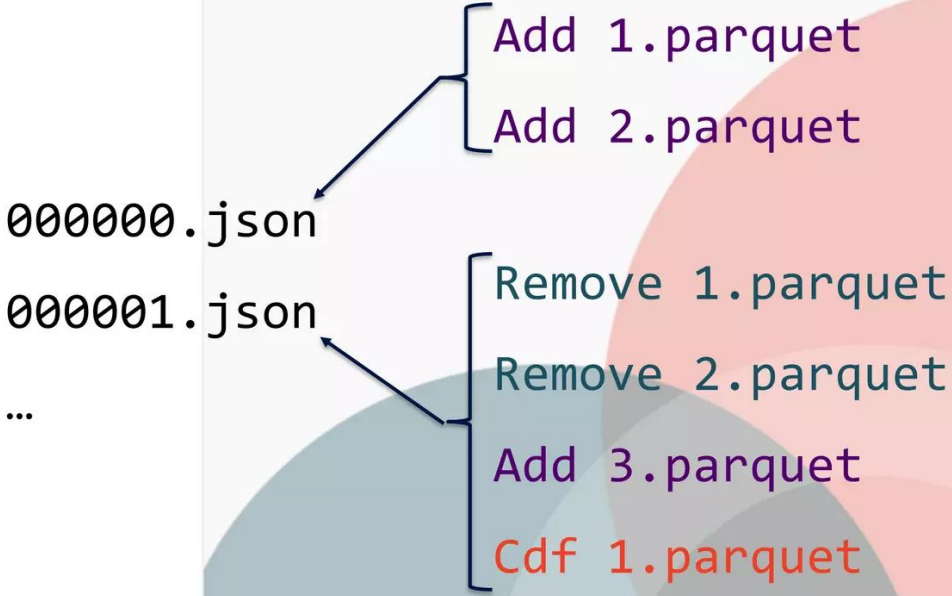
Storing the Delta Change Data Feed

Changes to the table are stored as ordered, atomic units called commits



Storing the Delta Change Data Feed

- When Change Data Feed is enabled, commits will reference an additional set of files containing the change data events
- These files contains the updated and deleted records



Storing the Delta Change Data Feed

- Silver & Gold Tables
  - Improve Delta performance by processing only data changes, and simplify ETL/ELT operations
- Materialized Views
  - Create up-to-date, aggregated views of information based on the data changes
- Transmit Changes
  - Send only the data changes to downstream systems
- Audit Trail Table
  - Capture and store all the data changes over time, including inserts, updates, and deletes

When to Use Delta Change Data Feed

- Delta changes include updates and/or deletes
- Small fraction of records updated in each batch
- Data received from external sources is in CDC format
- Send data changes to downstream application

- Delta changes are append only
- Most records in the table updated in each batch
- Data received comprises destructive loads
- Find and ingest data outside of the Lakehouse

Getting started with Delta Change Data Feed

Enable `CDF on TABLE`...

```
ALTER TABLE ...
SET TBLPROPERTIES
(delta.enableChangeDataFeed =
true);
```

in SQL, Python, or Scala

... or Establish `CDF on CLUSTER`

```
spark.conf.set('spark.databri
cks.delta.properties.defaults
.enableChangeDataFeed', True)
```

in Python, or Scala

Using Delta Change Data Feed

Query **changes**…

```
SELECT … FROM
table_changes('tableName',
    startingVersion
    [,endingVersion])
or
SELECT … FROM
table_changes('tableName',
    'startingTimestamp'
    [,'endingTimestamp'])
```

… and store **them**

```
INSERT INTO TABLE ...
USING delta ...
as
SELECT … FROM
table_changes(...)
```

in SQL, Python, or Scala

Getting started with Delta Change Data Feed

```
1  %sql
2  SELECT * FROM table_changes('silverTable', 2, 4) order by _commit_timestamp
```

▸ (1) Spark Jobs

	Country	NumVaccinated	AvailableDoses	_change_type	_commit_version	_commit_timestamp
1	Australia	100	3000	insert	2	2021-04-12T20:48:05.000+0000
2	USA	10000	20000	update_preimage	3	2021-04-12T20:48:08.000+0000
3	USA	11000	20000	update_postimage	3	2021-04-12T20:48:08.000+0000
4	UK	7000	10000	delete	4	2021-04-12T20:48:11.000+0000

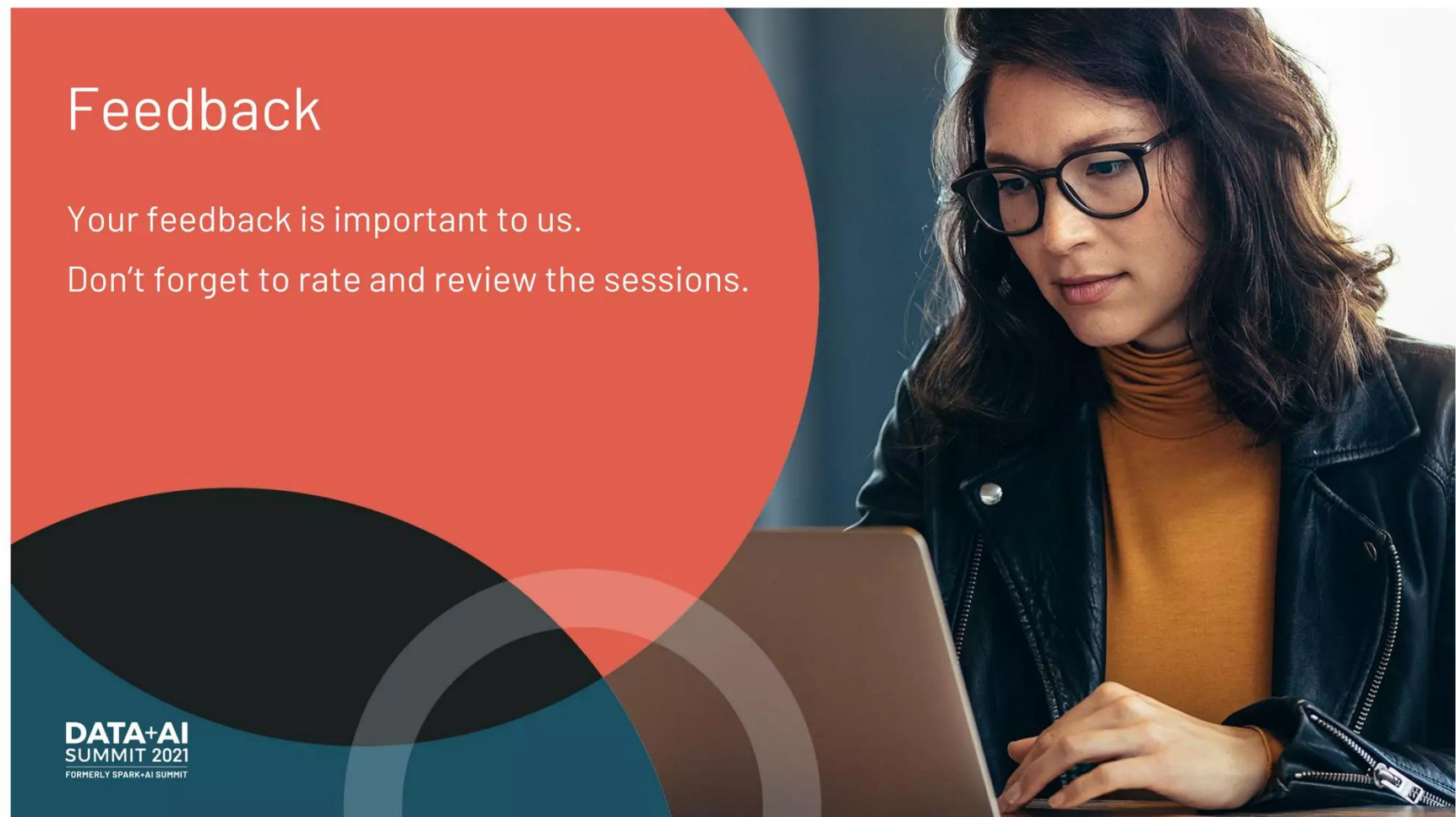
Showing all 4 rows.

Let’s look at some notebooks

Feedback

Your feedback is important to us.

Don't forget to rate and review the sessions.



DATA+AI SUMMIT 2021
FORMERLY SPARK+AI SUMMIT

DATA+AI SUMMIT 2021

FORMERLY SPARK+AI SUMMIT

ORGANIZED BY  databricks

#DataAISummit