Intro delta Lake

Wednesday, 11 January 2023 3:13 PM

ACID transactions (Atomicity, consistency, isolation, Durability).

→ These are properties that ensure data integrity, which is very important if you're dealing with transactions.

→ Firt data lake → 2010.

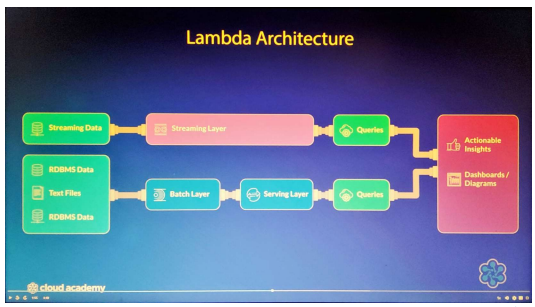
→ One problem of the data lake is they don't support ACID transactions and another problem is that its difficult to handle both batch and streaming data processing jobs.

→ Batch processing is normally performed on a large batch of data all at once, which can take a long time. In contrast, stream processing is performed continuosly as data streams in.

→ thus, batch processing provides historical data that's acurate but old and stream processing provides data that's incomplete but available immediately.

→ One attempt to combine both batch and stream is Lambda Arquitecture.

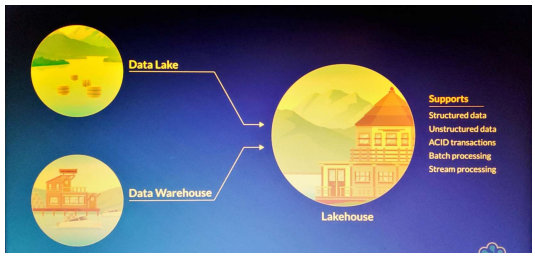
→ It uses separate pipelines for batch and stream processing, with different tools, each one, then they're queried separetely, and the data is combined into a unified report, by yet another system.

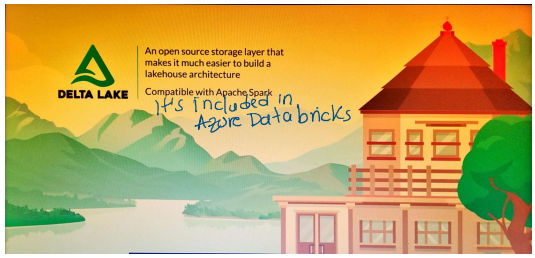


→ One of the biggest problem of Lambda Arq is complexity

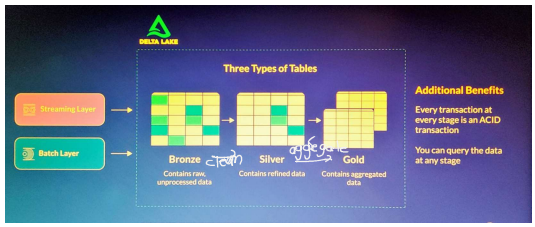
→ to solve this and many other problems, the Lakehouse was developed.

→ It's an attempt to combine the best elements of data warehouses and data lakes. It supports both structured and unstructured data, it support ACID transactions to ensure consistency, and supports





The way Delta lake suports both batch and streaming is by having a table format that can accommodate both types of data. that is, you can load the table. with historical batch data and then stream real-time data into it as well. Both types of data can be queried from the same table.



Using delta lake

Specify "delta" as the file format in your Apache Spark Code.

Delta file format consists of versioned Parquet files plus a transaction log.

→ Parquet is a format more efficient than Json or csv.

create table . . . using parquet → SQL.

format "delta"

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

→ create table in Python

```
dataframe = (spark.readStream
.format("delta")
.load(bronzeTable))
dataframe
.writeStream
.format("delta")
.option("checkpointLocation", SilverCheckpoint)
.outputMode("append")
.start(SilverTable)
```

→ Read/write streaming data.

→ saves the state, if it's interrupted, don't reinitialize, only continue.

→ no overwriting, only add to the table

Spark.streams.active → which streams are running

Features that are unique to Delta lake.

i) Ability to do upserts → it checks to see whether a previous version of the row already exists, if it doesn't, then the row is added. If it does, then the row is updated. (Data lakes don't support it)

```
MERGE INTO users
USING updates
ON users.userId = updates.userId
WHEN MATCHED THEN
UPDATE SET address = updates.addresses
WHEN NOT MATCHED THEN
INSERT (userId, address) VALUES (updates.userId,
updates.address)
```

the command to do an upsert is merge

Also, you can use merge to delete rows.

```
MERGE INTO users
USING opted_out_users
ON opted_out_users.userId = users.userId
WHEN MATCHED THEN DELETE
```

ii) time travel: Every time a delta table is updated, a new version is created, but the old version is still kept. then if you need to read data from a previous version, you can specify either the version ~~or~~ number or a timestamp

```
SELECT * FROM users
TIMESTAMP AS OF "2021-06-13"
WHERE userId = 111
```

→ VERSION AS OF...
→ with timestamp

```
%sql
DESCRIBE HISTORY customer_data_delta
```

→ to see the version #

(1) Spark Jobs

	version	timestamp	userId
1	3	2021-06-14T22:18:06.000+0000	3627391345911870
2	2	2021-06-14T22:17:36.000+0000	3627391345911870
3	1	2021-06-14T22:17:31.000+0000	3627391345911870
4	0	2021-06-14T22:17:26.000+0000	3627391345911870

```
RESTORE TABLE users TO
TIMESTAMP AS OF "2021-06-13"
```

→ Revert to a specific version

By default, delta tables only keep the commit history for 30 days.

To change that, you need to modify for the table.
- log Retention Duration
- deleted File Retention Duration → it says to VACUUM which files delete and which file it shouldn't

Optimizing delta lakes

There are 2 ways to make queries on a Delta Lake run faster.

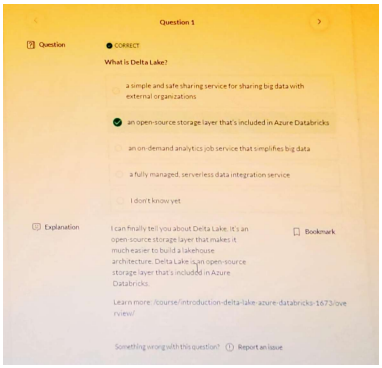
i) Partition the data properly (you want to distribute a table's data across partitions in such a way that queries will

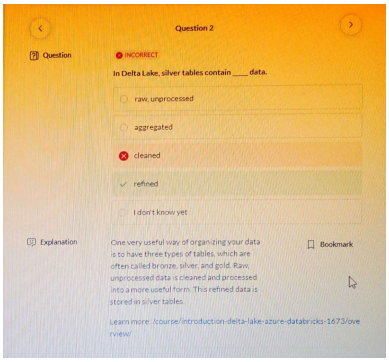
i) Partition the data properly (you want to distribute a table's data across partitions in such a way that queries will read data from many partitions, rather from just one or two. Reading data from many partitions at the same time takes advantage of parallelism to speed up the query. to distribute the table, you need to choose a good column to act as the partition key. (bad idea, columns with high cardinality, which means has a large number of unique values).
Partitions with the same size (for ex: 1GB)

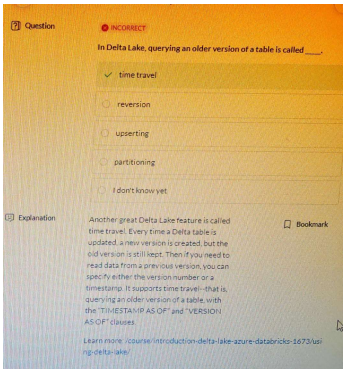
ii) To combine smaller files together into large files. when you write data to a Delta table, it'll store that data in one or more new files in each partition. It happens each time you write data to that table, so over time, you'll end up with a large number of files in each partition, this will slow down reads because it has to open so many files.
to compact these numerous small files into fewer large files, you can use OPTIMIZE command. or AUTO optimize option on the table. (it will do after every read)

Disadvantages to optimizing tables.

i) It uses a lot of resources, so it's an expensive operation.
ii) it slows down queries while it's running, this defeats the purpose of optimizing, you may schedule it to run when you're not executing queries that requires a fast response. time. tipically, OPTIMIZE is run once a day.
the optimization leaves all of those smaller files lying around. overtime, those files will take up a significant amount of space, so it would be best to get rid of them.
command VACUUM table name
However, this files are necessary if you want to revert a previous version.







Question

CORRECT

In Delta Lake, gold tables contain ____ data.

- cleaned
- refined, unaggregated
- raw, unprocessed
- aggregated
- I don't know yet

Explanation

Bookmark

One very useful way of organizing your data is to have three types of tables, which are often called bronze, silver, and gold. The refined data is aggregated into a form that would be suitable for business reporting, and it gets stored in gold tables.

Learn more: /course/introduction-delta-lake-azure-databricks-1673/overview/

Something wrong with this question? Report an issue

Question 3

Question

CORRECT

What command can be used to compact a large number of files in each partition into fewer large files?

- OPTIMIZE
- DEFRAG
- RESTORE
- VACUUM
- I don't know yet

Explanation

Bookmark

To compact a large number of files in each partition into fewer large files, you can use the OPTIMIZE command. Alternatively, you can enable the Auto Optimize option on a table.

Learn more: /course/introduction-delta-lake-azure-databricks-1673

Something wrong with this question? Report an issue