Merge Upsert in Azure Databricks: A Comprehensive Guide

In Azure Databricks, the Merge Upsert operation is a powerful way to synchronize target datasets with incoming data. It allows you to update existing records and insert new ones in a single operation, simplifying the data synchronization process.

This article will cover:

1. What is Merge Upsert?
2. Why use Merge Upsert?
3. Code demonstration with sample data
4. Before and after examples of the upsert operation

1. What is Merge Upsert?

Merge Upsert in Databricks is a combination of the MERGE SQL operation and the UPSERT logic. It merges two datasets based on a condition and either updates the existing records or inserts new records into the target dataset.

2. Why Use Merge Upsert?

Data Synchronization: It simplifies syncing data from different sources.

Efficiency: Single operation for both update and insert improves performance.

Versioning: Delta Lake (Databricks' storage format) allows versioning of data.

3. Code Demonstration with Sample Data

Let's walk through the MERGE UPSERT process using PySpark in Azure Databricks. We'll create two datasets: one for existing customer data and another for incoming updates. Then, we'll use the MERGE operation to upsert the data.

1. Prerequisites

Before we begin, ensure the following:

1. ADLS Gen2 Setup: You have Azure Data Lake Storage Gen2 set up with a container that holds the existing Delta table and incoming dataset.
2. Databricks and ADLS Connectivity: You have configured Databricks to connect to your ADLS Gen2 using either an Azure Service Principal or Azure Managed Identity.

2. Mount ADLS Gen2 in Databricks

First, mount your ADLS Gen2 storage to Databricks if it’s not already mounted.

```
# Example of mounting ADLS Gen2 using a service principal
configs = {
    "fs.azure.account.auth.type": "OAuth",
    "fs.azure.account.oauth.provider.type":
"org.apache.hadoop.fs.azurebfs.oauth2.ClientCredsTokenProvider",
    "fs.azure.account.oauth2.client.id": "<client-id>",
    "fs.azure.account.oauth2.client.secret": dbutils.secrets.get(scope="<scope-
name>", key="<secret-key>"),
    "fs.azure.account.oauth2.client.endpoint":
"https://login.microsoftonline.com/<tenant-id>/oauth2/token"
}

# Mount ADLS Gen2 container
dbutils.fs.mount(
    source = "abfss://<container-name>@<storage-account-
name>.dfs.core.windows.net/",
    mount_point = "/mnt/adls",
    extra_configs = configs
)
```

3. Load the Existing Data from ADLS Gen2

```
# Read existing customer data (stored as a Delta table in ADLS Gen2)
existing_data_path = "/mnt/adls/customers_delta/"
df_customers = spark.read.format("delta").load(existing_data_path)

# Show existing data
df_customers.show()
```

4. Load the Incoming Data from ADLS Gen2

```
# Read incoming updates data (e.g., stored as a CSV file in ADLS Gen2)
incoming_data_path = "/mnt/adls/incoming_data/customers_updates.csv"
df_incoming = spark.read.format("csv").option("header",
"true").option("inferSchema", "true").load(incoming_data_path)

# Show incoming data
df_incoming.show()
```

5. Perform the Merge (Upsert) Operation

```
from delta.tables import *

# Load the Delta table from ADLS Gen2
delta_table = DeltaTable.forPath(spark, existing_data_path)

# Perform Merge Upsert
delta_table.alias("customers").merge(
    df_incoming.alias("updates"),
    "customers.CustomerID = updates.CustomerID"  # Merge based on CustomerID
).whenMatchedUpdateAll(
).whenNotMatchedInsertAll(
).execute()
```

6. Verify the Results After the Upsert

```
# Read the merged data to verify the upsert
df_customers_merged = spark.read.format("delta").load(existing_data_path)
df_customers_merged.show()
```

Before Upsert: Existing Customer Data

Let's assume the existing customer dataset looks like this:

CustomerID	FirstName	LastName	Age
1	John	Doe	30
2	Jane	Smith	25
3	Sam	Williams	29
4	Tom	Harris	40
5	Emma	Johnson	22
6	Sophia	Brown	35
7	Olivia	Jones	28
8	James	Garcia	45
9	Liam	Martinez	32
10	Isabella	Davis	23

This dataset is already stored in ADLS Gen2 and read into Databricks using Delta.

```
df_customers.show()
```

The output:

CustomerID	FirstName	LastName	Age
1 2	John	Doe	30
3 4	Jane	Smith	25
5 6	Sam	Williams	29
7 8	Tom	Harris	40
9	Emma	Johnson	22
10	Sophia	Brown	35
	Olivia	Jones	28
	James	Garcia	45
	Liam	Martinez	32
	Isabella	Davis	23

Incoming Update Data

The incoming update dataset contains both new records and updates for existing customers.

CustomerID	FirstName	LastName	Age
2	Jane	Doe	26
4	Tom	Harris	41
5	Emma	Johnson	23
11	Noah	Wilson	38
12	Mia	Rodriguez	37

This incoming data is read from ADLS Gen2.

```
df_incoming.show()
```

CustomerID	FirstName	LastName	Age
2	Jane	Doe	26
4	Tom	Harris	41
5	Emma	Johnson	23
11	Noah	Wilson	38
12	Mia	Rodriguez	37

3. After Upsert: Updated Customer Data

After the upsert operation, the Delta table now contains both the updated and newly inserted records. Let's load the updated Delta table to check the results.

```
df_customers_merged = spark.read.format("delta").load("/mnt/adls/customers_delta")
df_customers_merged.show()
```

The output:

CustomerID	FirstName	LastName	Age	
1	John	Doe	30	
2	Jane	Doe	26	#Updated
3	Sam	Williams	29	
4	Tom	Harris	41	#Update
5	Emma	Johnson	23	d
6	Sophia	Brown	35	
7	Olivia	Jones	28	#Update
8	James	Garcia	45	
9	Liam	Martinez	32	d
10	Isabella	Davis	23	
11	Noah	Wilson	38	# New
12	Mia	Rodriguez	37	# New

Comparison of Before and After Upsert

Before Upsert

CustomerID	FirstName	LastName	Age
1 2	John	Doe	30
3 4	Jane	Smith	25
5 6	Sam	Williams	29
7 8	Tom	Harris	40
9	Emma	Johnson	22
10	Sophia	Brown	35
	Olivia	Jones	28
	James	Garcia	45
	Liam	Martinez	32
	Isabella	Davis	23

After Upsert

CustomerID	FirstName	LastName	Age
1 2	John	Doe	30
3 4	Jane	Doe	26
5 6	Sam	Williams	29
7 8	Tom	Harris	41
9	Emma	Johnson	23
10	Sophia	Brown	35
11	Olivia	Jones	28
12	James	Garcia	45
	Liam	Martinez	32
	Isabella	Davis	23
	Noah	Wilson	38
	Mia	Rodriguez	37