**Azure Data Factory Incremental Data Load with Change Tracking - Customer Table Use Case**

**1. Objective:**

This document outlines the steps to implement incremental data load using SQL Server's **Change Tracking** feature for the **customer\_data** table. The goal is to capture only the changed records (inserts, updates, deletes) from the customer\_data table and load them into a destination table.

**2. Database Objects and Operations:**

**a. Source Table: customer\_data**

Create the new customer\_data table to store customer information.

sql

CREATE TABLE customer\_data

(

CustomerID INT NOT NULL,

CustomerName VARCHAR(255),

ContactNumber VARCHAR(50),

Address VARCHAR(500),

PRIMARY KEY (CustomerID)

);

This table will be the source for incremental data loads.

**b. Change Tracking Version Table: change\_tracking\_version**

Create the table change\_tracking\_version to store the current version of Change Tracking for the customer\_data table.

sql

CREATE TABLE change\_tracking\_version

(

TableName VARCHAR(255),

SYS\_CHANGE\_VERSION BIGINT

);

**c. Stored Procedure: Update\_ChangeTracking\_Version**

The stored procedure Update\_ChangeTracking\_Version will update the Change Tracking version after each data load.

sql

CREATE PROCEDURE Update\_ChangeTracking\_Version

@CurrentTrackingVersion BIGINT,

@TableName VARCHAR(50)

AS

BEGIN

UPDATE change\_tracking\_version

SET [SYS\_CHANGE\_VERSION] = @CurrentTrackingVersion

WHERE [TableName] = @TableName;

END

**3. Step-by-Step Operations:**

**Step 1: Enable Change Tracking on Database and Table**

Enable Change Tracking at both the database level and the table level for customer\_data.

sql

-- Enable Change Tracking for the database

ALTER DATABASE your\_database\_name

SET CHANGE\_TRACKING = ON

(CHANGE\_RETENTION = 2 DAYS, AUTO\_CLEANUP = ON);

-- Enable Change Tracking for the `customer\_data` table

ALTER TABLE customer\_data

ENABLE CHANGE\_TRACKING

WITH (TRACK\_COLUMNS\_UPDATED = ON);

* **CHANGE\_RETENTION = 2 DAYS**: Retains change data for 2 days.
* **AUTO\_CLEANUP = ON**: Automatically removes outdated change data.

**Step 2: Insert Initial Data into customer\_data Table**

Insert initial customer data into the customer\_data table.

sql

INSERT INTO customer\_data (CustomerID, CustomerName, ContactNumber, Address)

VALUES

(1, 'John Doe', '555-1234', '123 Elm St, Springfield'),

(2, 'Jane Smith', '555-5678', '456 Oak St, Rivertown'),

(3, 'Alice Johnson', '555-8765', '789 Pine St, Laketown'),

(4, 'Bob Brown', '555-4321', '101 Maple St, Uptown'),

(5, 'Charlie Davis', '555-6789', '202 Birch St, Downtown');

**Step 3: Track Change Version**

Track the initial version of Change Tracking and store it in the change\_tracking\_version table.

sql

DECLARE @ChangeTrackingVersion BIGINT;

SET @ChangeTrackingVersion = CHANGE\_TRACKING\_CURRENT\_VERSION();

-- Store the current change tracking version for `customer\_data`

INSERT INTO change\_tracking\_version

VALUES ('customer\_data', @ChangeTrackingVersion);

**Step 4: Modify Data in customer\_data Table (Insert and Update)**

To simulate data changes, insert a new record and update an existing record.

sql

-- Insert new data into `customer\_data` table

INSERT INTO customer\_data (CustomerID, CustomerName, ContactNumber, Address)

VALUES (6, 'David Wilson', '555-1122', '303 Cedar St, Midtown');

-- Update existing data in `customer\_data` table

UPDATE customer\_data

SET ContactNumber = '555-9999', CustomerName = 'Johnathan Doe'

WHERE CustomerID = 1;

**Step 5: Validate the Change Tracking Version**

Validate the Change Tracking version to ensure it has been correctly updated.

sql

SELECT \* FROM change\_tracking\_version;

At this point, the change\_tracking\_version table should contain the most recent tracking version for customer\_data.

**Step 6: Update Change Tracking Version After Data Load**

After processing the incremental data load using Azure Data Factory, update the Change Tracking version using the Update\_ChangeTracking\_Version stored procedure.

sql

DECLARE @CurrentTrackingVersion BIGINT;

SET @CurrentTrackingVersion = CHANGE\_TRACKING\_CURRENT\_VERSION();

-- Update the change tracking version for `customer\_data`

EXEC Update\_ChangeTracking\_Version @CurrentTrackingVersion, 'customer\_data';

**Step 7: Clean Up (Optional)**

Optionally, drop the test tables and the stored procedure after testing.

sql

-- Clean-up (drop the tables and stored procedure)

DROP TABLE customer\_data;

DROP TABLE change\_tracking\_version;

DROP PROCEDURE Update\_ChangeTracking\_Version;

**4. Azure Data Factory Pipeline Design**

The above SQL steps can be automated using **Azure Data Factory (ADF)** for incremental data loads. Below are the key steps in the pipeline:

**Step 1: Lookup Activity**

The **Lookup Activity** in ADF will query the change\_tracking\_version table to get the current SYS\_CHANGE\_VERSION for the customer\_data table.

* **SQL Query**:

sql

SELECT SYS\_CHANGE\_VERSION FROM change\_tracking\_version WHERE TableName = 'customer\_data';

**Step 2: Copy Data Activity**

The **Copy Data Activity** will use the CHANGES function to select only the rows that have changed since the last tracked version.

* **SQL Query**:

sql

SELECT CustomerID, CustomerName, ContactNumber, Address

FROM customer\_data

WHERE SYS\_CHANGE\_VERSION > @CurrentTrackingVersion;

This ensures that only the modified rows (insert, update, delete) since the last version are copied to the destination.

**Step 3: Stored Procedure Activity**

After the incremental data load, use the **Stored Procedure Activity** in ADF to call the Update\_ChangeTracking\_Version stored procedure and update the Change Tracking version.

* **SQL Command**:

sql

DECLARE @CurrentTrackingVersion BIGINT;

SET @CurrentTrackingVersion = CHANGE\_TRACKING\_CURRENT\_VERSION();

EXEC Update\_ChangeTracking\_Version @CurrentTrackingVersion, 'customer\_data';