**Azure Data Factory Incremental Data Load with Change Tracking – Employees Table**

**1. Objective:**

To implement incremental data load using SQL Server's **Change Tracking** feature. The goal is to capture only the changed records (inserts, updates, deletes) from the source table employee\_data and load them into a destination table. We will also create the required stored procedures and other database objects for tracking changes.

**2. Database Objects and Operations:**

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

First, we will create a new table employee\_data to store employee information.

sql

CREATE TABLE employee\_data

(

EmployeeID INT NOT NULL,

EmployeeName VARCHAR(255),

Department VARCHAR(100),

Salary DECIMAL(18, 2),

PRIMARY KEY (EmployeeID)

);

This table will serve as the source for incremental data loads.

**b. Change Tracking Version Table: CTV\_Employees**

We will create a table CTV\_Employees to store the current version of Change Tracking for the employee\_data table.

sql

CREATE TABLE CTV\_Employees

(

TableName VARCHAR(255),

SYS\_CHANGE\_VERSION BIGINT

);

**c. Stored Procedure: SP\_Update\_CTV\_Employees**

The stored procedure SP\_Update\_CTV\_Employees will update the version of Change Tracking for the employee\_data table after each data load.

sql

CREATE PROCEDURE SP\_Update\_CTV\_Employees

@CurrentTrackingVersion BIGINT,

@TableName VARCHAR(50)

AS

BEGIN

UPDATE CTV\_Employees

SET [SYS\_CHANGE\_VERSION] = @CurrentTrackingVersion

WHERE [TableName] = @TableName;

END

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

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

To enable Change Tracking, we first need to enable it at the database level, and then at the table level for employee\_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 `employee\_data` table

ALTER TABLE employee\_data

ENABLE CHANGE\_TRACKING

WITH (TRACK\_COLUMNS\_UPDATED = ON);

* **CHANGE\_RETENTION = 2 DAYS**: Retains changes for 2 days.
* **AUTO\_CLEANUP = ON**: Automatically cleans up the Change Tracking data.

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

Now, we insert some sample data into the employee\_data table.

sql

INSERT INTO employee\_data (EmployeeID, EmployeeName, Department, Salary)

VALUES

(1, 'John Doe', 'Engineering', 75000),

(2, 'Jane Smith', 'Marketing', 65000),

(3, 'Alice Johnson', 'HR', 58000),

(4, 'Bob Brown', 'Finance', 90000),

(5, 'Charlie Davis', 'IT', 72000);

**Step 3: Track Change Version**

We will track the current version of Change Tracking and store it in the CTV\_Employees table.

sql

DECLARE @ChangeTrackingVersion BIGINT;

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

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

INSERT INTO CTV\_Employees

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

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

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

sql

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

INSERT INTO employee\_data (EmployeeID, EmployeeName, Department, Salary)

VALUES (6, 'David Wilson', 'Operations', 80000);

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

UPDATE employee\_data

SET Salary = 77000, EmployeeName = 'Johnathan Doe'

WHERE EmployeeID = 1;

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

To ensure that Change Tracking is enabled and working, you can validate the current version.

sql

SELECT \* FROM CTV\_Employees;

At this point, the CTV\_Employees table should contain the most recent tracking version for employee\_data.

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

After performing the incremental data load using Azure Data Factory, we will update the Change Tracking version using the stored procedure SP\_Update\_CTV\_Employees.

sql

DECLARE @CurrentTrackingVersion BIGINT;

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

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

EXEC SP\_Update\_CTV\_Employees @CurrentTrackingVersion, 'employee\_data';

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

After performing the operations, you can clean up by dropping the created tables and stored procedure for testing purposes.

sql

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

DROP TABLE employee\_data;

DROP TABLE CTV\_Employees;

DROP PROCEDURE SP\_Update\_CTV\_Employees;

**4. Azure Data Factory Pipeline Design**

The above SQL steps will be implemented in an **Azure Data Factory (ADF)** pipeline for automating the incremental data load. Here are the key steps in the pipeline:

**Step 1: Lookup Activity**

The **Lookup Activity** in ADF will retrieve the current SYS\_CHANGE\_VERSION from the CTV\_Employees table.

* **SQL Query**:

sql

SELECT SYS\_CHANGE\_VERSION FROM CTV\_Employees WHERE TableName = 'employee\_data';

**Step 2: Copy Data Activity**

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

* **SQL Query**:

sql

SELECT EmployeeID, EmployeeName, Department, Salary

FROM employee\_data

WHERE SYS\_CHANGE\_VERSION > @CurrentTrackingVersion;

This query ensures that only the rows with a newer SYS\_CHANGE\_VERSION than the last one processed are copied to the destination.

**Step 3: Stored Procedure Activity**

After the incremental data load, we use a **Stored Procedure Activity** to call the SP\_Update\_CTV\_Employees stored procedure and update the tracking version.

* **SQL Command**:

sql

DECLARE @CurrentTrackingVersion BIGINT;

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

EXEC SP\_Update\_CTV\_Employees @CurrentTrackingVersion, 'employee\_data';