Workshop 9

Part 3

Implementing Change Data Capture (CDC) Using Apache Nifi

Overview:

Change Data Capture (CDC) is a widely used data integration pattern designed to **detect and track changes** in source systems and **notify downstream systems** that depend on this data. By monitoring inserts, updates, and deletes, CDC ensures data consistency across multiple systems and enables timely responses to changes.

In today's data-driven enterprises, real-time awareness of data changes - such as new transactions, customer updates, or order processing - is essential. CDC plays a vital role in ensuring that applications and services across the organization stay synchronized and up to date.

Workshop Objective

In this part of the workshop, you'll implement a CDC pipeline using **Apache NiFi** to monitor a **MySQL table** in real time. Upon detecting an **insert operation**, NiFi will transform the change event into **Avro format** and write it to **HDFS**.

Prerequisites:

Before you begin, complete the following setup:

1. Create a **MySQL database** named: `cdc`
2. Inside this database, create a **table** named: `employee`
3. Open the NiFi dataflow named:
   Workshop 9 - `CDC - MySQL Events to HDFS Avro`

Dataflow description:

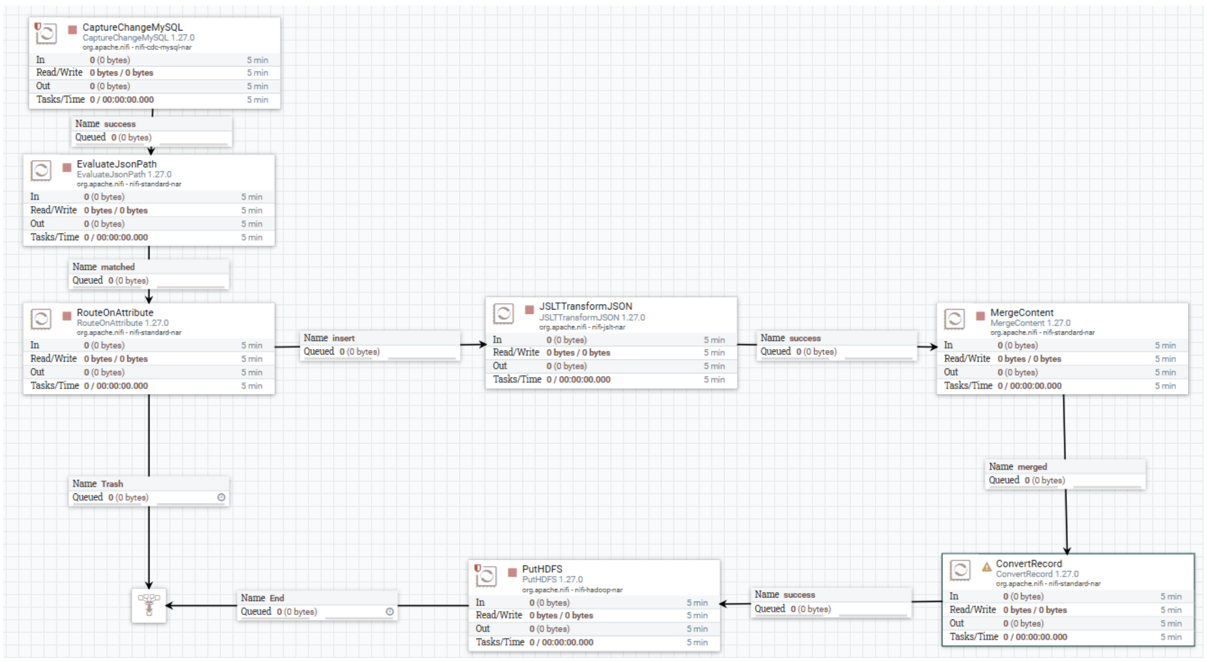
The dataflow consists of **7 processors** and **3 controller services**, and it performs the following steps:

Step-by-Step Flow:

1. **Monitor the MySQL table** in real time. (For this workshop, we are focusing only on **insert operations**.)
2. When an insert occurs, NiFi generates a **FlowFile** containing the data change in JSON format.
3. The dataflow evaluates the operation type and filters to retain only the **insert** events.
4. A **JSLT/JOLT transformation** is applied to clean and reshape the JSON message.

5. The refined JSON is then **converted to Avro format**.
6. The Avro records are **merged and written to HDFS**.

Nifi Dataflow Overview:



Processor Overview.

Processor Name	Role Description
`CaptureChangeMySQL`	Listens to changes on the MySQL employee table and emits change notifications in JSON format.
`EvaluateJsonPath`	Extracts the operation type (insert/update/delete) from the JSON payload..
`RouteOnAttribute`	Filters out all operations except 'insert'.
`JSLTTransformJSON`	Applies a transformation to restructure the JSON payload. JSLT Transformation Used : `{ for (.columns) .name : .value }`
`MergeContent`	Combines multiple FlowFiles into a single output to prevent writing small files to HDFS.

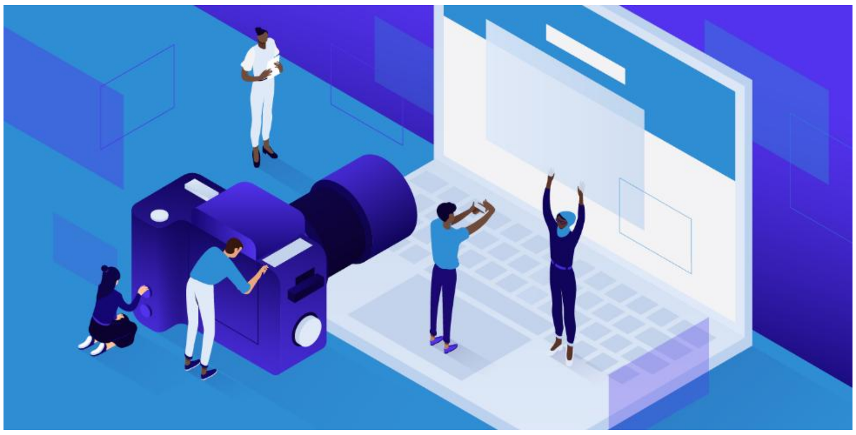
`ConvertRecord`	Converts the cleaned JSON into Avro format using a schema.
`PutHDFS`	Stores the resulting Avro files into HDFS. Output directory: `/workshops/nifi/cdc/employee`

Online Resources

To learn more about JSON transformations with **JOLT** and **JSLT**, refer to the resources below:

- **Cloudera Quick Reference for NiFi JOLT Processors**
  https://community.cloudera.com/t5/Community-Articles/Jolt-quick-reference-for-Nifi-Jolt-Processors/ta-p/244350
- **JOLT Reference and Examples**
  https://intercom.help/godigibee/en/articles/4044359-transformer-getting-to-know-jolt
- **Online JOLT Playground**
  https://jolt-demo.appspot.com/#inception
- **JSLT GitHub Repository**
  https://github.com/schibsted/jslt

Time to take your snapshot.



And Shutdown your Sandbox to free the allocated resources.