syncsort

DMX-h ETL Use Case Accelerators

File CDC

© Syncsort® Incorporated, 2015

All rights reserved. This document contains proprietary and confidential material, and is only for use by licensees of DMExpress. This publication may not be reproduced in whole or in part, in any form, except with written permission from Syncsort Incorporated. Syncsort is a registered trademark and DMExpress is a trademark of Syncsort, Incorporated. All other company and product names used herein may be the trademarks of their respective owners.

The accompanying DMExpress program and the related media, documentation, and materials ("Software") are protected by copyright law and international treaties. Unauthorized reproduction or distribution of the Software, or any portion of it, may result in severe civil and criminal penalties, and will be prosecuted to the maximum extent possible under the law.

The Software is a proprietary product of Syncsort Incorporated, but incorporates certain third-party components that are each subject to separate licenses and notice requirements. Note, however, that while these separate licenses cover the respective third-party components, they do not modify or form any part of Syncsort's SLA. Refer to the "Third-party license agreements" topic in the online help for copies of respective third-party license agreements referenced herein.

Table of Contents

1 Introduction 1

2 File CDC with DMX-h IX 2

2.1 CDC_SingleTarget Task 3

2.2 CDC_MultiTarget Task 4

Appendix A File CDC with DMX-h User-defined MapReduce 5

A.1 Map Step 5

A.1.1 MT_SortData.dxt 6

A.2 Reduce Step 7

A.2.1 RT_SplitSides.dxt 8

A.2.2 RT_PerformJoinCDC_SingleOutput.dxt 9

A.2.3 RT_PerformJoinCDC.dxt 10

1 Introduction

Change data capture (CDC) is a process in which previous and current versions of a large data set are compared to determine the changes between the two versions. This use case accelerator implements CDC in DMX-h ETL by performing a full-outer join on the previous and current versions of a large file.

The use case accelerators are developed as standard DMExpress jobs with just minor accommodations that allow them to be run in or outside of Hadoop:

- When specified to run in the Hadoop cluster, DMX-h Intelligent Execution (IX) automatically converts them to be executed in Hadoop using the optimal Hadoop engine, such as MapReduce. In this case, some parts of the job may be run on the Linux edge node if not suitable for Hadoop execution.
- Otherwise, they are run on a Windows workstation or Linux edge node, which is useful for development and testing purposes before running in the cluster.

While the method presented in the next section is the simplest and most efficient way to develop this example, the more complex user-defined MapReduce solution is provided as a reference in Appendix A.

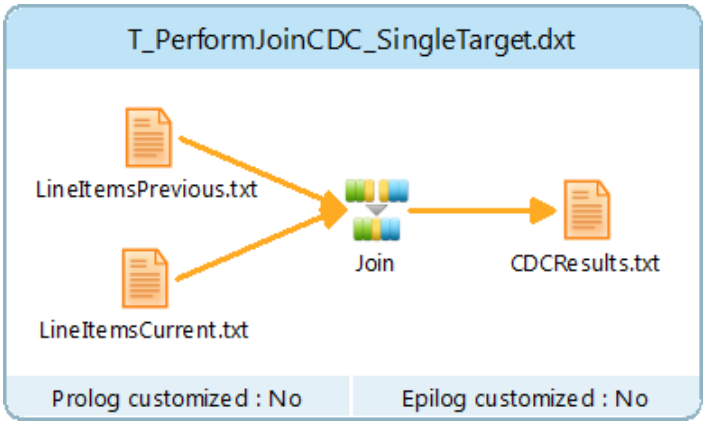
For guidance on setting up and running the examples both outside and within Hadoop, see Guide to DMX-h ETL Use Case Accelerators.

For details on the requirements for IX jobs, see “Developing Intelligent Execution Jobs” in the DMExpress Help.

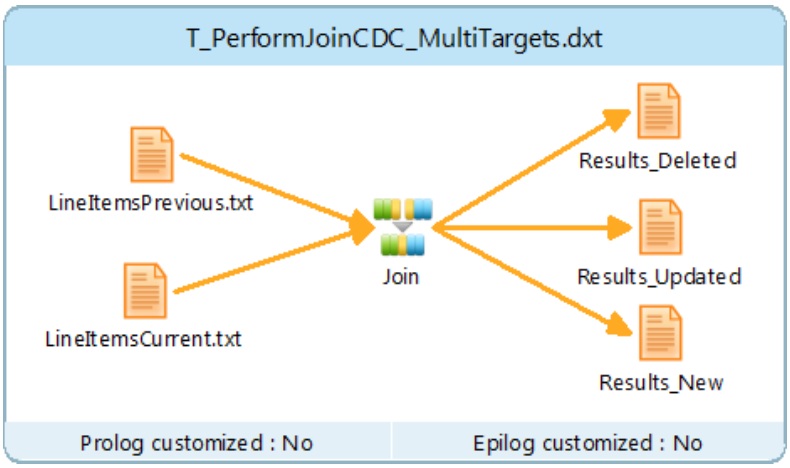
2 File CDC with DMX-h IX

The CDC solution in DMX-h ETL consists of a job with only one join task. There are two different approaches: single-target, shown in J_FileCDC.dxj, and multi-target, shown in J_FileCDC_MultiTargets.dxj. Both versions join two large files, and vary only in the type of output they produce.

- Single-target File CDC produces a single target file that contains all changed records, with a flag added to each record indicating whether it is an insert, delete, or update.



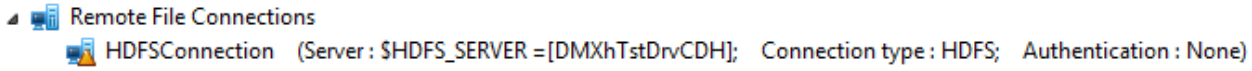
- Multi-target File CDC produces three separate targets, one each for inserted (new), deleted, and updated records.



In both cases, inserts, deletes, and updates are determined as follows:

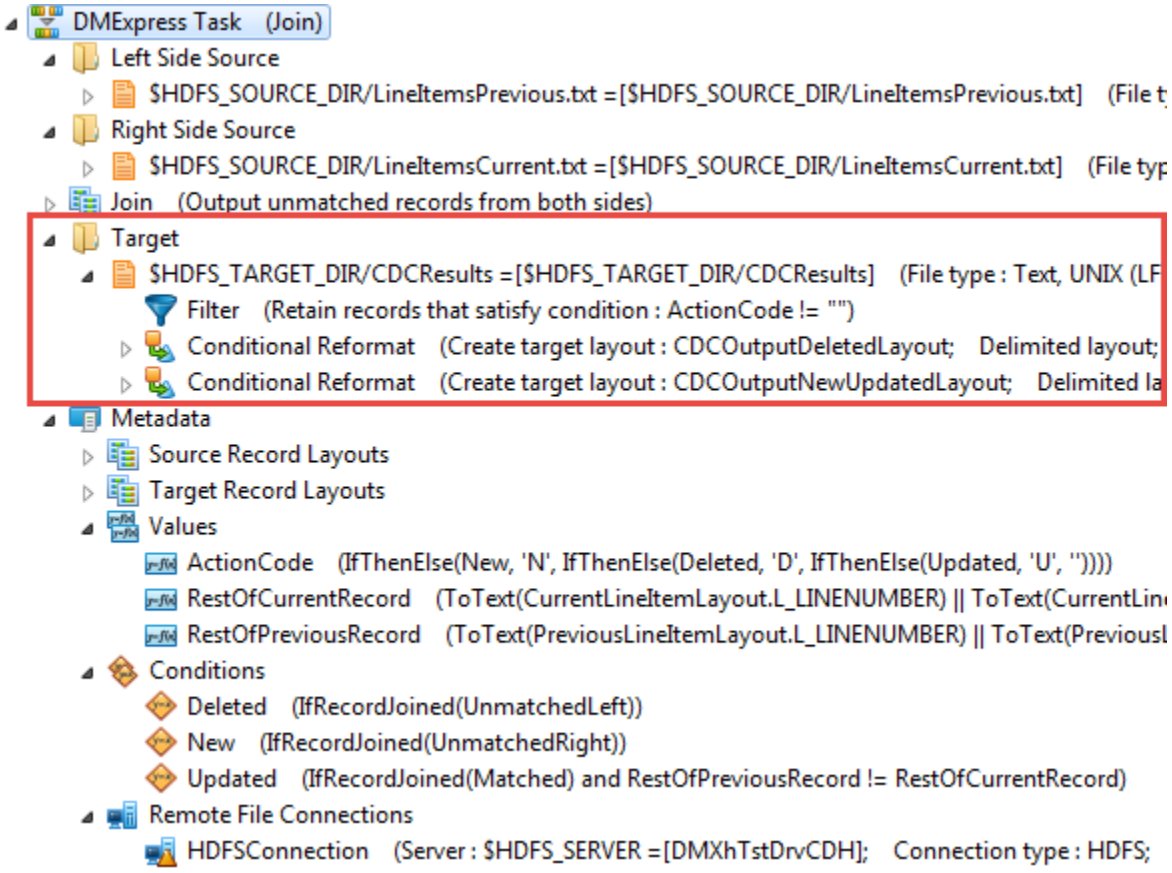
- Unmatched records in the current version are inserts.
- Unmatched records in the previous version are deletes.
- Matched records in both the current and previous versions whose non-primary-key fields are not identical are updates.

Also in both, the HDFS source and target files are defined with a remote file connection to the Hadoop cluster:



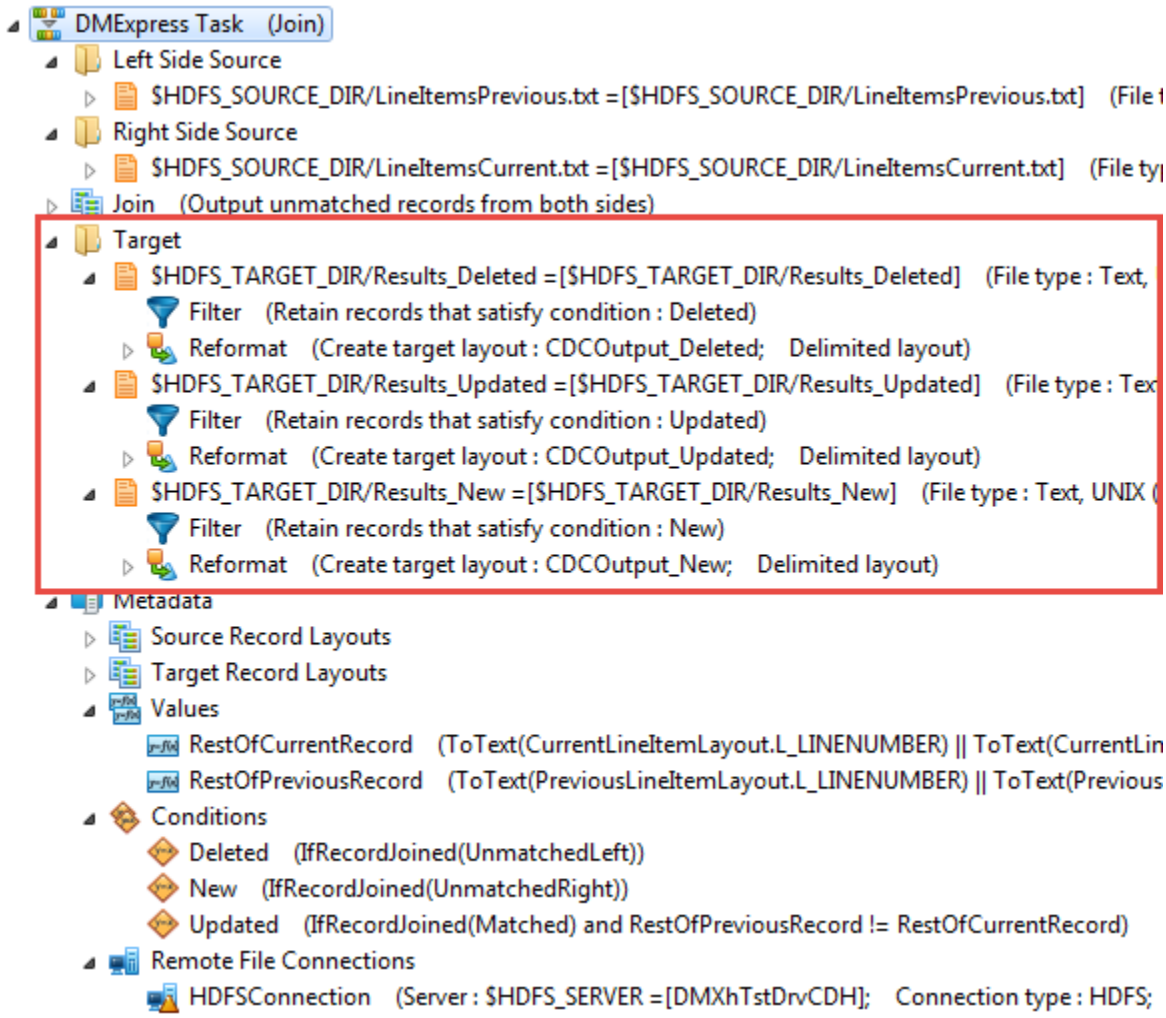
2.1 CDC_SingleTarget Task

In the single-target version, the final target of the join task is a single file containing all changed records, with each record flagged with an ActionCode to indicate whether it is new ('N'), deleted ('D'), or updated ('U'). Unchanged records have a blank ActionCode and are excluded from the output using a target filter.



2.2 CDC_MultiTarget Task

In the multi-target version, the final targets of the join task are three files, one each for new, deleted, and updated records:



Appendix A File CDC with DMX-h User-defined MapReduce

This user-defined MapReduce solution is provided as a reference in the event that particular knowledge of your application's data would benefit from manual control of the MapReduce process.

The File CDC job in user-defined MapReduce contains a map step and a reduce step, separated by a MapReduce data flow connector between the final map task and the initial reduce task, as follows:

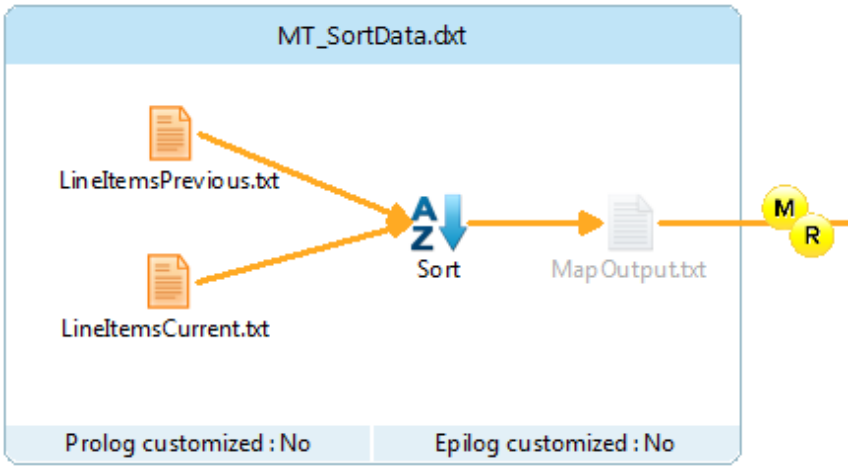
The map step reads two input files and produces a single output stream, marking each record to denote which source file it came from. It then assigns partition IDs to the data such that all records with the same sort key, regardless of source file, will go to the same reducer.

The reduce step splits the single stream of data back out into two separate files, then performs the join on the set of records it received. One version of the reduce step writes all new, updated, and deleted records to a single target, marking each record to denote its change status, and the other version writes the three record types to three separate targets.

While it may seem like wasted work to have the mapper merge the records together only to split them back apart in the reducer, this is necessary because in a MapReduce job, only a single stream of data can be passed between mappers and reducers, and a map-only job would not work because each mapper only receives a subset of the data, which may not include all the records with a given sort key. However, since each mapper assigns partition IDs to the records in the same way, it guarantees that all records with the same sort key will end up in the same reducer so that the join will capture all the changes.

A.1 Map Step

The File CDC map step in DMX-h consists of one task which marks each input record with its side ID, assigns a partition ID to each record based on the key, sorts on that partition ID, and outputs the records in a single stream.



While a join requires two inputs, only a single data path exists between the mappers and the reducers in the Hadoop environment. All records from both inputs are thus combined into a single data stream by the mappers, with a side ID added to each record so the reducers can identify which side the record came from, and a partition ID added to the beginning of each record to direct records with the same key to the same reducer, as shown in the following sample input and output data.

Sample source data records:

LineItemsPrevious.txt:

1,155190,7706,1,17,21168.23,0.04,0.02,N,O,1996-03-13, …

LineItemsCurrent.txt:

1,15635,638,6,32,49620.16,0.07,0.02,N,O,1996-01-30, …

3,19036,6540,2,49,46796.47,0.1,0,R,F,1993-11-09, …

Sample single-stream output from map step, with partition ID and side ID added at the beginning:

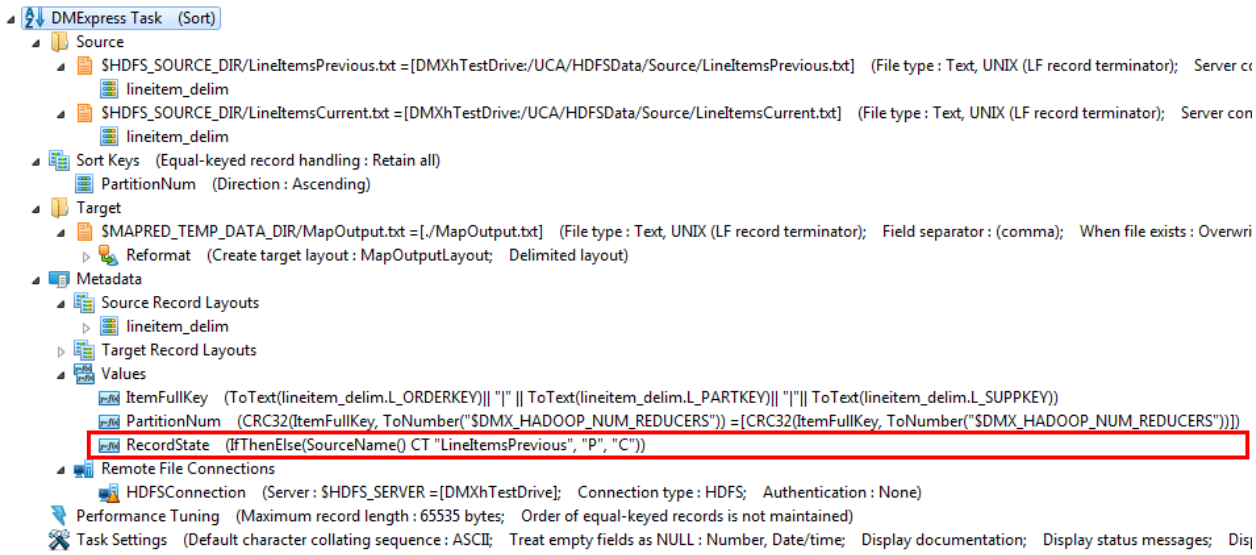
0,P,1,155190,7706,1,17,21168.23,0.04,0.02,N,O,1996-03-13, …

0,C,1,15635,638,6,32,49620.16,0.07,0.02,N,O,1996-01-30, …

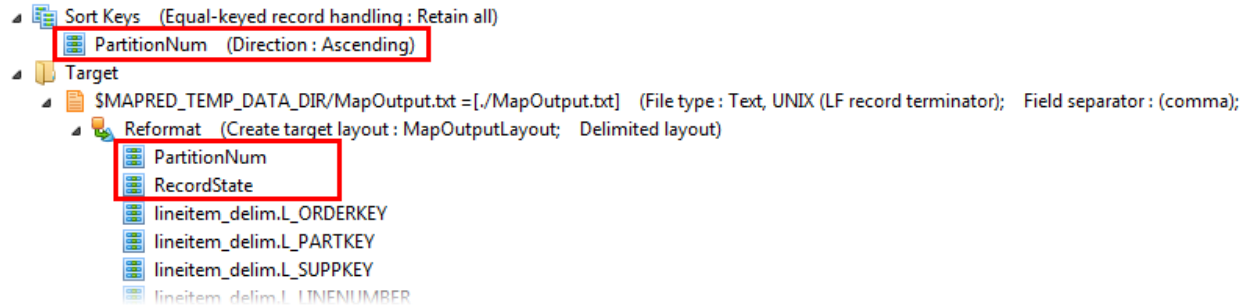
0,C,3,19036,6540,2,49,46796.47,0.1,0,R,F,1993-11-09, …

A.1.1 MT_SortData.dxt

This task reads both source files and assigns the side IDs to the records based on the source file names. The task uses the SourceName() function to retrieve the source file name. If the file name contains the text “LineItemPrevious”, the record is marked with a “P” for previous, otherwise, it is marked with a “C” for current.



The task further assigns the partition ID (PartitionNum) as the first field in each target record and sorts all records by that value so that the framework can properly distribute the data to the correct reducer.

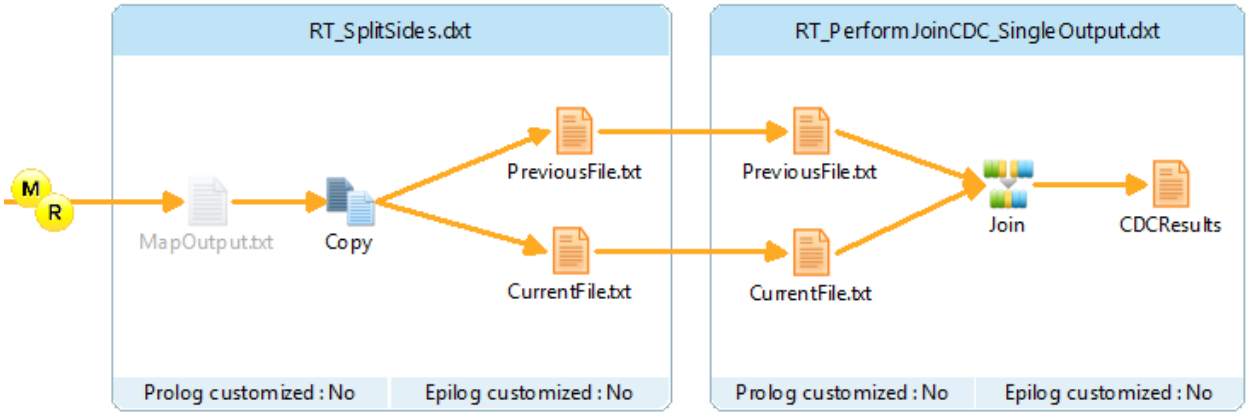


Since all records with the same primary key value (L_ORDERKEY + L_PARTKEY + L_SUPPKEY) must go to the same reducer, the partition ID is determined by creating a CRC32() hash value based on this key. In addition, the environment variable DMX_HADOOP_NUM_REDUCERS (automatically set to the configured number of reducers when running in Hadoop) is passed to the function to limit the range of partition ID values to the number of reducers invoked for your MapReduce job.

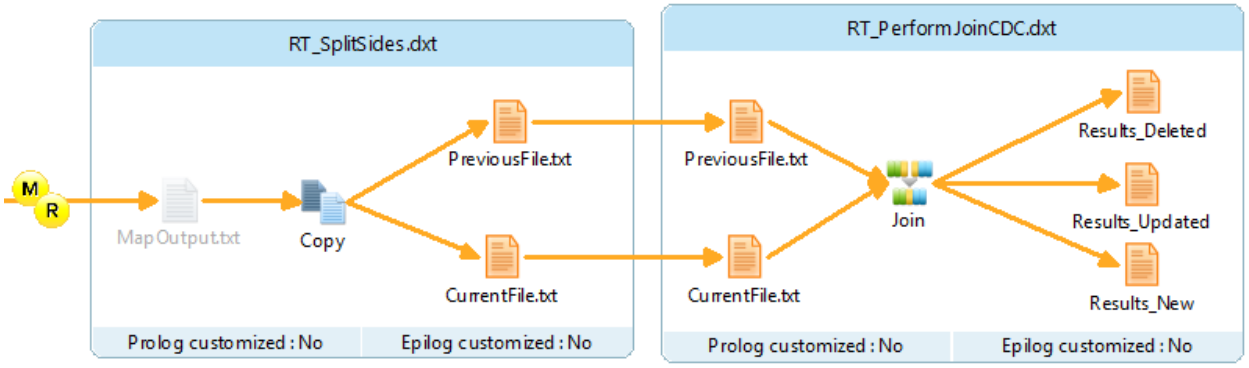
A.2 Reduce Step

The File CDC reduce step in DMX-h ETL consists of one task to split the data back out into separate files, and one to perform the join and output the results. There are two versions of the second task, one which writes to a single target, and one which writes to three separate targets.

In the single target version, the output of the join task goes to a single target, with each record flagged to indicate whether it was added, deleted, or updated.



In the multi-target version, the output of the join task goes to three separate targets, one each for added, deleted, and updated records.



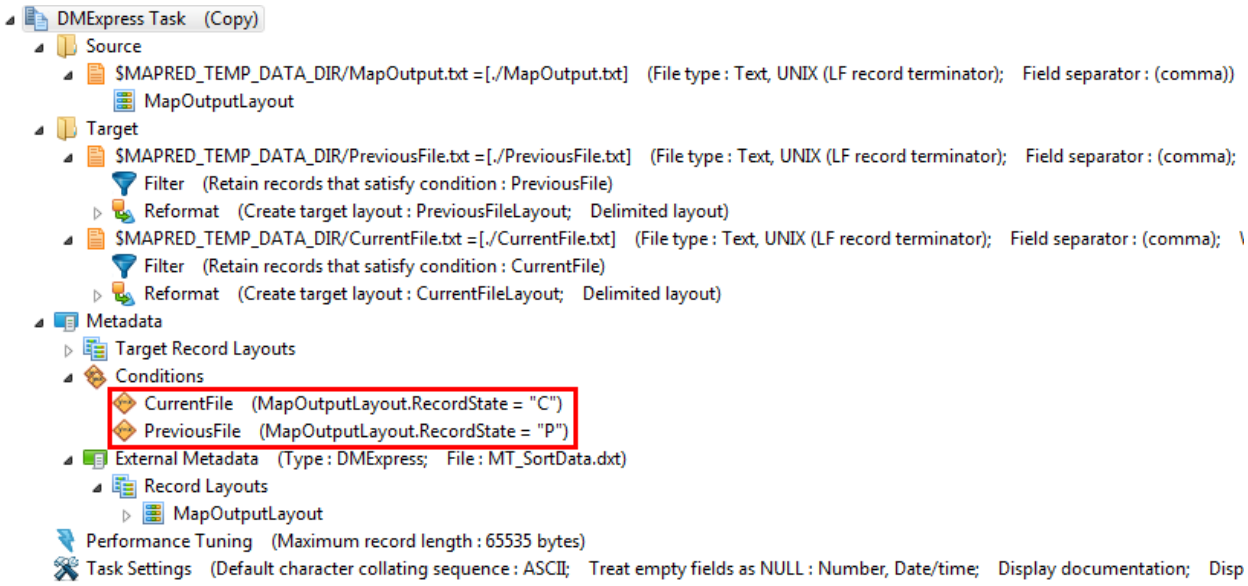
In both cases, inserts, deletes, and updates are determined as follows:

- Unmatched records in the current version are inserts.
- Unmatched records in the previous version are deletes.
- Matched records in both the current and previous versions whose non-primary-key fields are not identical are updates.

As the partition IDs were assigned by the map job based on the primary key values, previous and current versions of records that contain equal primary key values are routed to the same reducer, ensuring correct CDC processing.

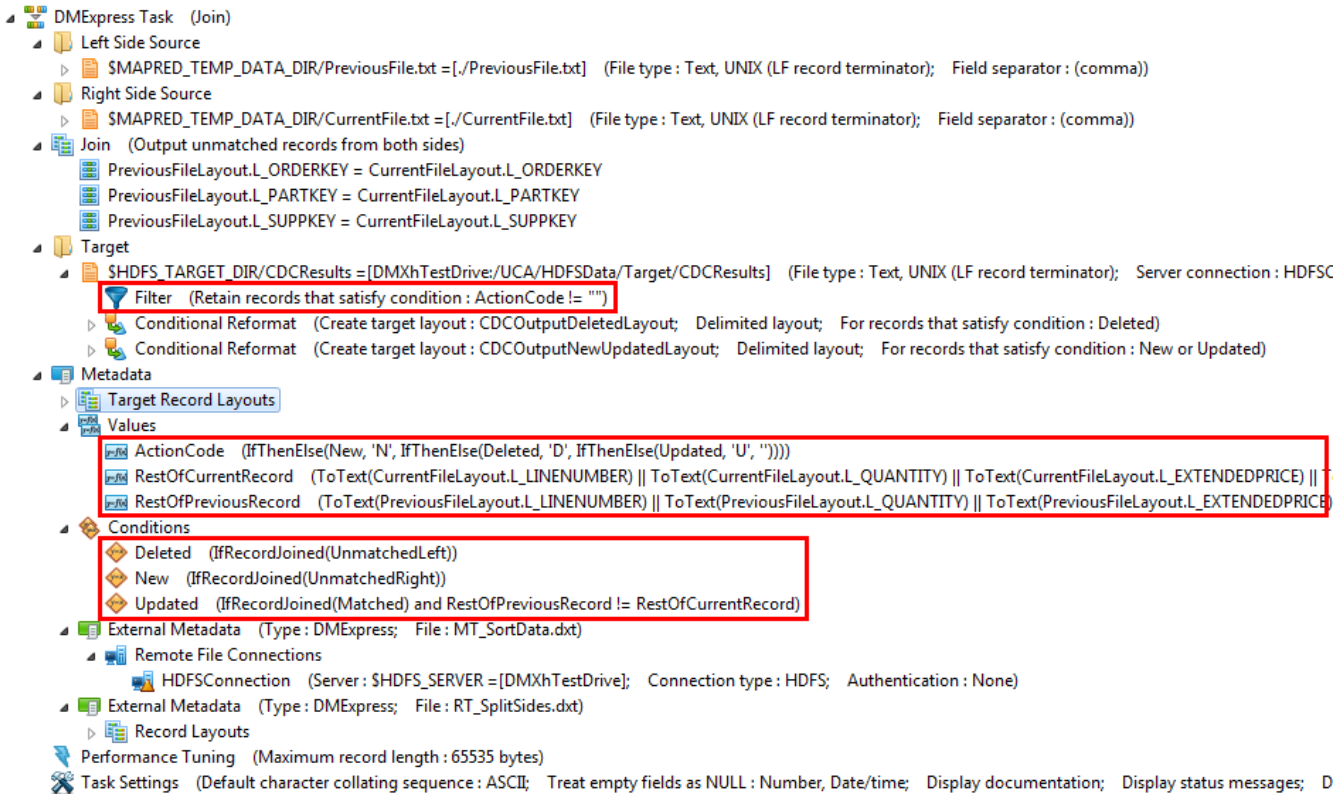
A.2.1 RT_SplitSides.dxt

This task reads the single input stream containing one partition of data generated by the mappers. Based on the side ID, it separates the data into two temporary files, PreviousFile.txt and CurrentFile.txt.



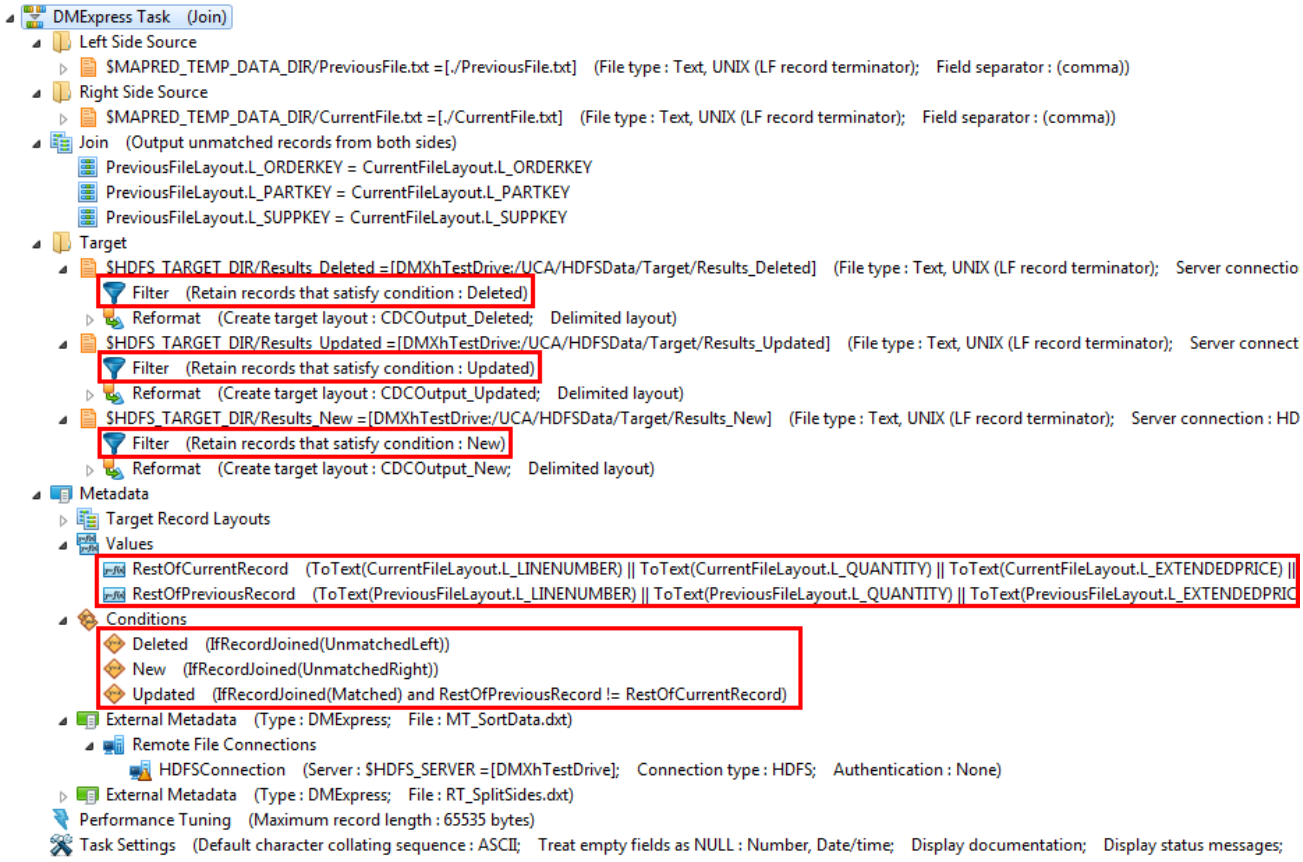
A.2.2 RT_PerformJoinCDC_SingleOutput.dxt

This task joins the temporary files into one large file and writes the results to a single target, where each record is marked with an ActionCode value: ‘N’ for new; ‘D’ for deleted, or ‘U’ for updated. Unchanged records have a blank ActionCode and are excluded from the output using a target filter. The Hadoop framework will create a directory with the target name specified here, and each reducer’s output will be written to its own “part-reducer_ID” file within that directory.



A.2.3 RT_PerformJoinCDC.dxt

An alternative to the previous task, this task joins the temporary files and writes the results to three targets, one each for new, deleted, and updated records. Since Hadoop can only write one target and this task has multiple targets, DMX-h writes them directly to HDFS, but uses a similar “part” naming convention to Hadoop, as described previously.



About Syncsort

Syncsort provides enterprise software that allows organizations to collect, integrate, sort, and distribute more data in less time, with fewer resources and lower costs. Thousands of customers in more than 85 countries, including 87 of the Fortune 100 companies, use our fast and secure software to optimize and offload data processing workloads. Powering over 50% of the world's mainframes, Syncsort software provides specialized solutions spanning "Big Iron to Big Data", including next gen analytical platforms such as Hadoop, cloud, and Splunk. For more than 40 years, customers have turned to Syncsort's software and expertise to dramatically improve performance of their data processing environments, while reducing hardware and labor costs. Experience Syncsort at www.syncsort.com.

Syncsort Inc. ***50 Tice Boulevard, Suite 250, Woodcliff Lake, NJ 07677*** ***201.930.8200***