Data Migrator for i
User's Guide
Part 2

IBM Db2 Web Query for i

DataMigrator ETL Extension

June 2021

Contents Part 2

1 Scheduling and e-mail

1.1 Overview

When performing an ETL process, an important requirement may be to run it in a recurring, automated fashion. For example, the process must run once a day, normally at night, to capture the day's changes and incorporate them into a data warehouse. Another requirement may be getting a confirmation, usually via email, that the ETL process was successful or get notified if a problem occurred.

DataMigrator provides a scheduling ability through the DMC. This scheduling tool is separate from Web Query'sreporting scheduler and RUNWQFEX command contained in Scheduler and Standard Editions, , allowing it to run independently of any scheduled reports. In addition, DataMigrator provides email notification for key steps in the ETL process, such as when a process flow completes successfully or fails. It should be noted that RUNWQFEX CL command can be used to run a flow but that is not covered here (refer to the Db2 Web Query wiki documents at http://ibm.biz/db2wqwiki).

> **NOTE**: The DataMigrator server, which is the Web Query server, must be active for scheduling to work. You can check if the Db2 Web Query server is active by using the `WRKWEBQRY` CL command to verify the status is Active and all ports are active.

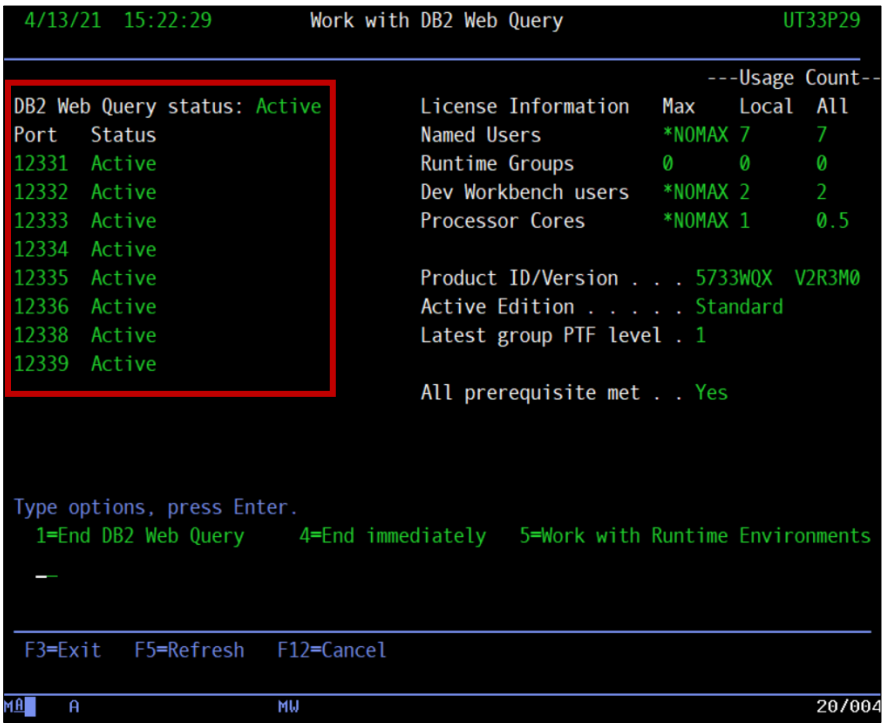


Figure 1 Output of WRKWEBQRY showing all ports are active

1.2 Scheduling setup

To schedule a flow, right click on the flow and click `Schedule and E-mail`, then `Manage`. Do this for `flow02`.

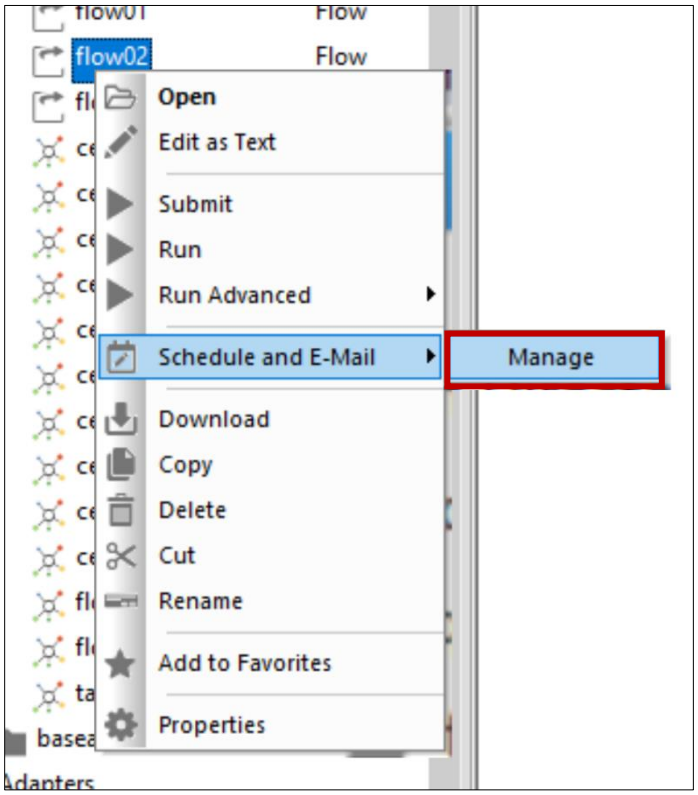


Figure 2 Scheduling flow02

We can setup the scheduling details for the flow from the scheduling screen. In the Scheduling Status dropdown choose `Active`.

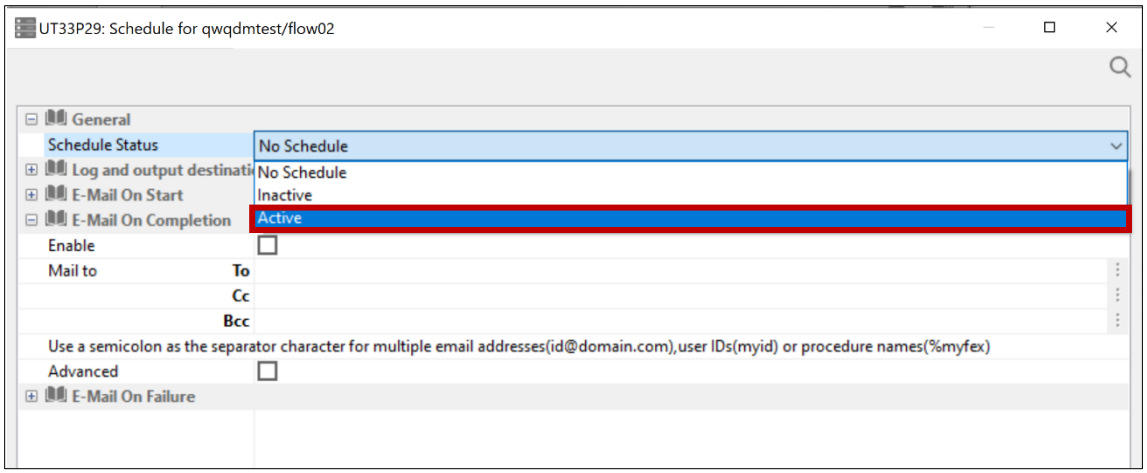


Figure 3 Setting schedule status to active

Once the Schedule Status is `Active`, a Scheduling Type option appears. Select `Multi-Day` from the dropdown. The Start Date and Start Time are pre-populated with the current date and time. Change the start time to `00:00`, which represents midnight. We could specify an ending date, but we will leave it unchecked.

Under Special date/time ranges we can choose from different days of the week and of the month. We will schedule the flow to run every other day of the week, on the 15th of each month (middle of the month) and on the last day of the month.

Click the ellipsis (…) on Days of the Week, which will bring up a window with days of the week. Select a day by clicking on it. Deselect a selected day by clicking it again. Select `Sunday`, `Tuesday`, `Thursday` and `Saturday`. Then click `OK`.

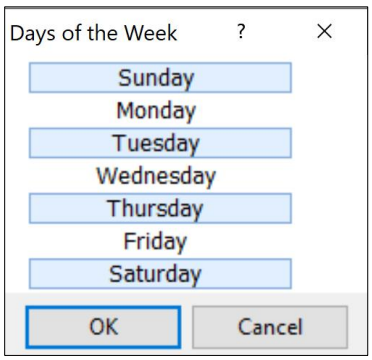


Figure 4 Days of the week selected

Click on the Days of the Month ellipsis (…), which will bring up a window with days of the month and a special value `Last Day of Month`. As before, select a day by clicking on it. Deselect a selected day by clicking it again. Select `15` and `Last Day of Month`. Then click `OK`.

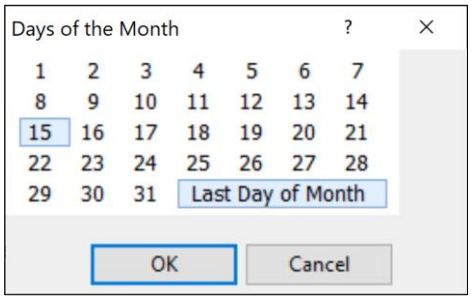


Figure 5 Days of the month selected

Back on the main schedule screen, verify the scheduled dates are shown and then click `Set` to activate the schedule.

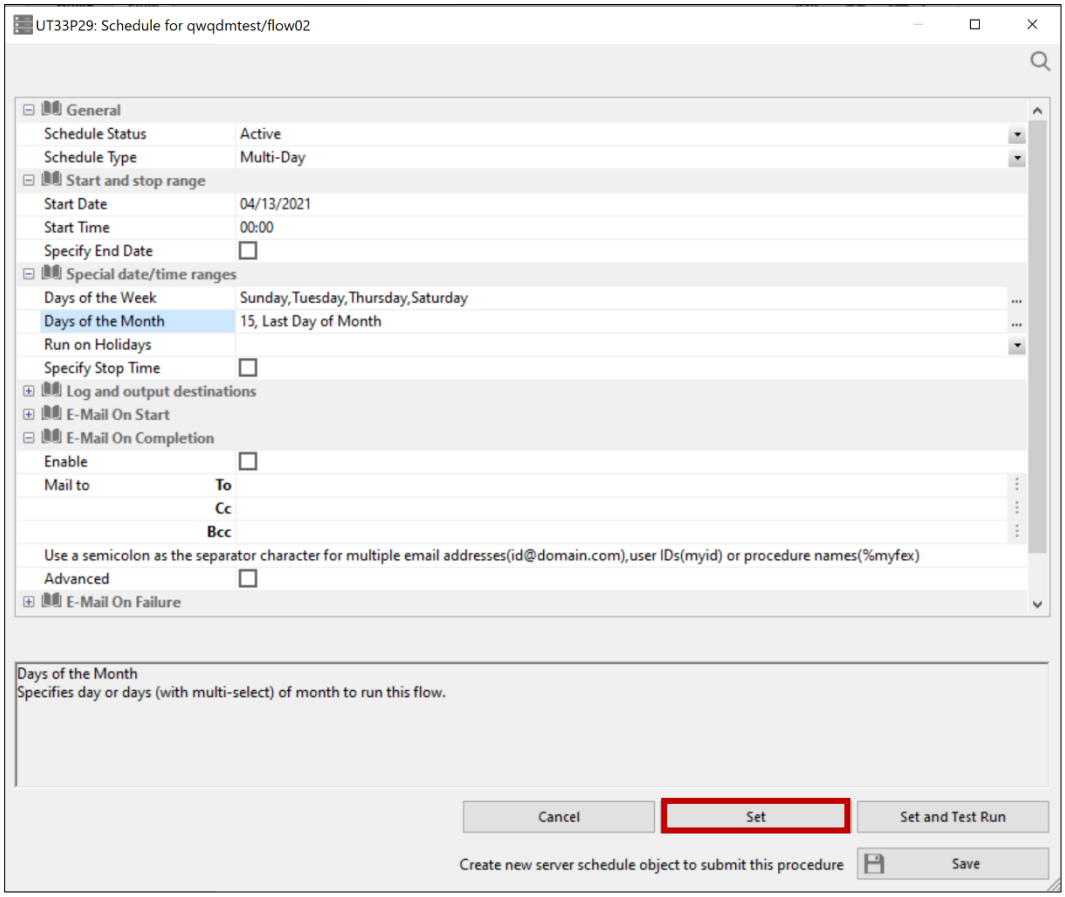


Figure 6 Verifying the scheduled dates

A verification screen is displayed. Click `OK`.

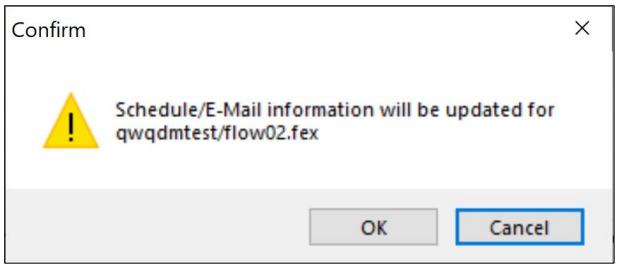


Figure 7 Schedule enablement verification

Looking closely at the icon associated with `flow02` in the left navigation tree, we see that it has changed to a clock, indicating it is a scheduled flow. Now `flow02` is scheduled to run.

Figure 8 Scheduled flow

1.3 E-mail notification setup

DataMigrator provides several built-in e-mail options for scheduling. An email can be sent when a scheduled flow starts, when it completes, or when it fails. Depending on your needs, you may want to know every time it starts and runs successfully or only when it fails. We will setup to be informed only when the scheduled flow fails.

Open the scheduling screen again by right clicking on `flow02` and clicking `Schedule and E-mail`, then `Manage`.

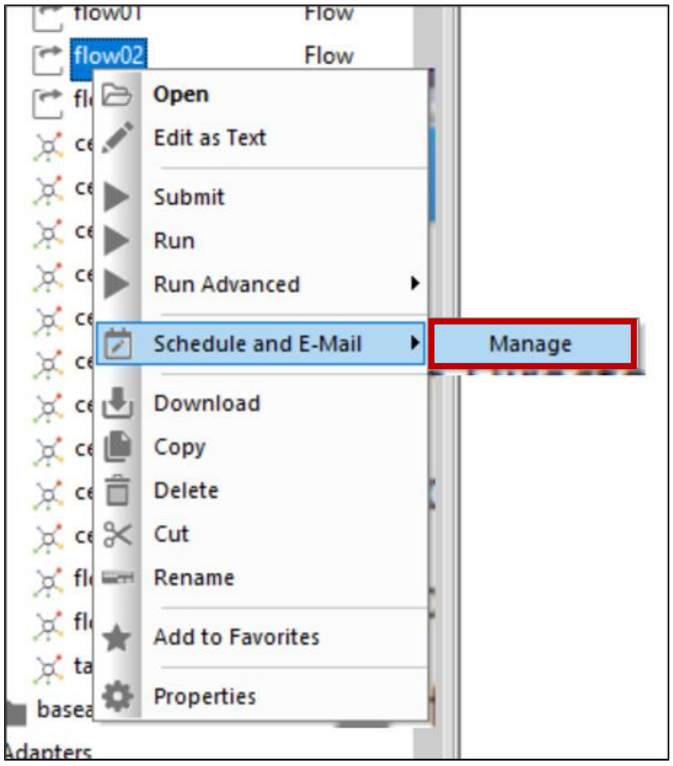


Figure 9 Scheduling flow02

E-mail control options are shown at the bottom of the screen. Expand the E-mail on failure section. Click the checkbox next to `Enable`. In the `Mail to` option, type in your email address in `To`. Click the checkbox next to `Advanced` to show more options for setting the email Importance, a subject line, and the email body. Set the Importance to `High`. Set e-mail Subject to `'flow02 did not complete successfully!'`. Set e-mail message to `'Nightly data flow flow02 ended with a failure on system xxxx'` where `xxxx` is your system name.

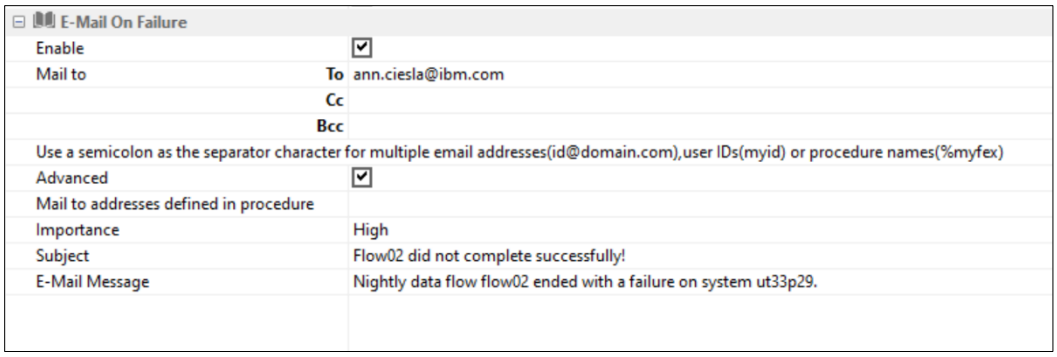


Figure 10 Setup for receiving an email on failure

Verify the options and click `Set` to activate. Click `OK` on the confirmation screen.

We now have flow02 set to run four days a week, plus on the 15th and last day of each month, and to send us an email if something fails.

> **WARNING:** Since this was a test flow, you likely do not really want it to run on your system. Therefore, before leaving this chapter, bring up the schedule again and change the Schedule Status to `Inactive`. This will preserve all the schedule settings but will avoid running the flow.

2 Using Change Data Capture (CDC)

2.1 Overview

Change Data Capture (CDC) enables reading from *journals* to determine what rows in an associated table (file) have changed. This approach is very useful for capturing maintenance changes or 'trickle feed' ETL from large source tables where it would take too long to process the entire tables.

To use Change Data Capture, the file must be configured to use journaling. A *Table Log Records* synonym is created to represent a log 'table', allowing reading from the journal entries and representing them as records from the selected table. In other words, the Table Log Records synonym is really a synonym for a journal but has columns that match the associated table being journaled. This table log synonym can be used as a source in a data flow or in a direct load flow.

It is important to note that the IBM i operating system allows you to set up journal receivers to be deleted from the system automatically. But this could occur before a CDC flow executes and captures the changed data. Therefore, it is *not* recommended to delete journal receivers automatically. This means that you must 'clean up' your system by manually deleting old journal receivers.

You can also set up journal receivers to minimize entry data, meaning that when a change occurs the system only records the fields that changed. However, CDC cannot process minimized entry data and requires a copy of the entire record. So, when setting up a journal receiver for CDC, minimize entry data should be set to `*NONE`.

An important aspect to understand about DataMigrator CDC is that it works on a transaction basis. When processing entries from a journal, CDC only processes transactions that have been committed. For example, if a database job is writing records to a journaled table and CDC is setup against the table's journal, none of the records written into the table (and therefore the journal) will be processed by CDC until the database job issues a `COMMIT`. Once the `COMMIT` entry makes it to the journal, then CDC will read the record images from the journal and process them.

Remote journaling, a feature of IBM i, can also be leveraged in a CDC architecture with DataMigrator. Remote journaling can be very useful to minimize any impact to the production or source system you are wanting to source changed data from by allowing the OS to copy entries in the journal receiver from one system/LPAR to another. DataMigrator can pull the changed data from the "remote" journal receiver that is actually local to where DataMigrator is running.

2.2 Definitions

The first four columns in a log synonym are added automatically by DataMigrator to every Table Log Record synonym. They are used for flow control during DataMigrator IUD processing. The columns have the following functions:

- *CDC_OPER* – the operation type (Insert, Update, or Delete)
- *CDC_TID* – the Transaction ID
- *CDC_TIMES* – the time stamp
- *CDC_RRN* – RRN of the record

The Configuration section of the CDC synonym also consists of special parameters with properties that can be customized. Each CDC parameter in the configuration section of the CDC synonym has a default value of the global variable. Each global variable name begins with `&&CDC_` followed by the parameter name.

The following options are available:

2.2.1 DB Log Parameters

- *DATA_ORIGIN* – this is set to `DBMSLOG` to indicate the origin of the data is a DBMS table log

- *START* – indicates the starting point for reading journal entries, known as 'log records'. Possible values are:
  - `CHKPT` – after the last transaction retained in the checkpoint file. This is the default.
  - `CUR_TRAN` – the first transaction in the journal after the job started.
  - `CUR_LOG` – the first available transaction in the current active journal receiver.

- *CHKPT_SAVE* – indicates if the last processed transaction should be saved in a checkpoint file. Possible values are:
  - `YES` – retain the last processed transaction in the checkpoint file.
  - `NO` – do not retain the last processed transaction. This option aides testing because using it with `Sample Data` or `Test Transformations` will not reset the checkpoint.

- *CHKPT_*FILE – indicates the location and name of the file used to store checkpoint information. Possible values are:
  - `Blank` – if no file name is specified, a file is created with the same name as the synonym with the extension `chp`. This is the default.
  - `Physical location` – the full directory and file name in the format for the operating system used. This option should be used when there are multiple flows that will access the table log synonym for the same table. Each flow should have a unique synonym with a unique name for the checkpoint file.

    - `Application directory` – the file is created in the same application directory as the synonym.

- *COMMIT_MODE* – supports the transaction commitment control. Possible values are:
    - `ON` – normal transaction mode with a commit used. This is the default.
    - `OFF` – no commit issued, or the DBMS uses auto-commit mode.
    - `DTC` – distributed transaction mode is used.

- *LOG_NAME* – indicates the name of the journal. If this is blank, the default is `QSQJRN` which is the name of the journal associated with SQL tabled.

- *LOG_LOCATION* – indicates the journal library. If this is blank, the default is the same library as the associated database table.

2.2.2 Listening Parameters

- *POLLING* – indicates the polling interval in seconds, or how often the database log is scanned. The default is 1 second.

- *TIMEOUT* – indicates the timeout interval in seconds. If there is no activity in this time interval, the processing will stop. A value of zero means there is no limit. The default value is 1 second.

Note that the two options above work together. For example, if *POLLING* is 2 and *TIMEOUT* is 10, the server polls the log every 2 seconds for new transactions. If there are no new transactions for a 10 consecutive second timespan, then polling stops.

2.2.3 Read Limits

- *MAXLUWS* – indicates the maximum number of database transactions (Logical Units of Work or LUWS) to process before stopping the job. A value of zero (0) means all transactions. The default value is 1 transaction.

2.3 CDC for Local Journals

2.3.1 Setup

We will need a table to use as a data source that can also be changed by adding new records. We will copy the inventory table from `QWQCENT` into the `QWQDMTEST` library. Using SQL, issue the following statements. Make sure to be running under some level of commitment control, for example `*CHG`.

> **NOTE**: To run under commitment control through Access Client Solutions (ACS) Run SQL Scripts, click `Connection`, then click `Connected – xxx` where `xxx` is your system and click `JDBC Settings`. Then change the `Isolation level`.

```
CREATE TABLE qwqdmtest.inventory as
(select * from qwqcent.inventory) WITH DATA;
COMMIT;
```

Figure 11 Creating a table for inventory

A synonym for a journal allows reading from the journal associated with a selected table. The synonym contains the same columns as the table itself plus four additional columns that identify changes to the table. However, a synonym for a journal is *not* a substitute for a synonym for the table itself. A synonym is still needed for the table to be able to read or write from the table directly. So, we need two synonyms.

First, create a normal synonym for our new `qwqdmtest.inventory` table. Remember to specify adapter of `*LOCAL`. Give it a prefix of `dmtest_`.

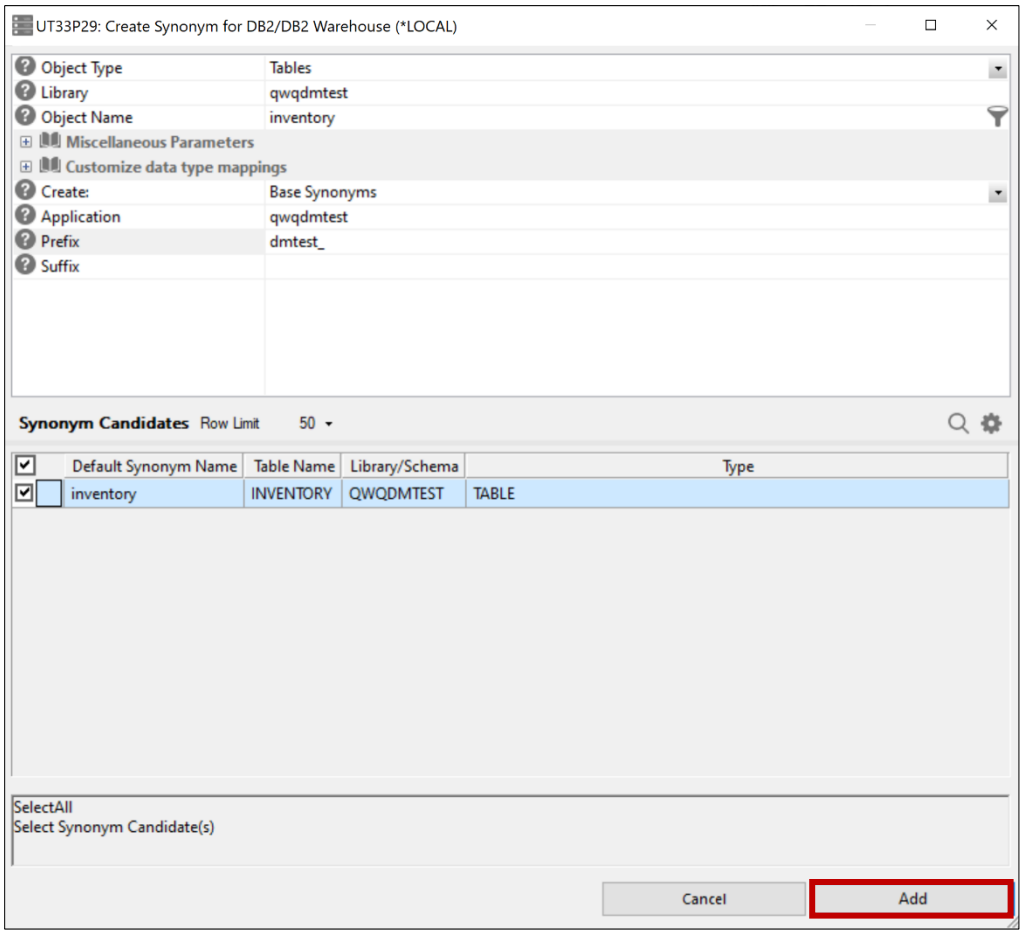


Figure 12 Creating a synonym for qwqdmtest.invetory

Next, we need to create the Table Log Records synonym for the journal. Right click an application directory in the navigation pane and click `New`, then `Synonym`. Choose the `*LOCAL` adapter. From the Object Type dropdown menu, select `Table Log Records`. Specify the library `qwqdmtest` and table name `inventory`. Click the checkbox next to `inventory`. Use the prefix `dmtest_` and the suffix `_log`. The suffix will be a reminder that this synonym reads from a log (journal). Then click `Add`.

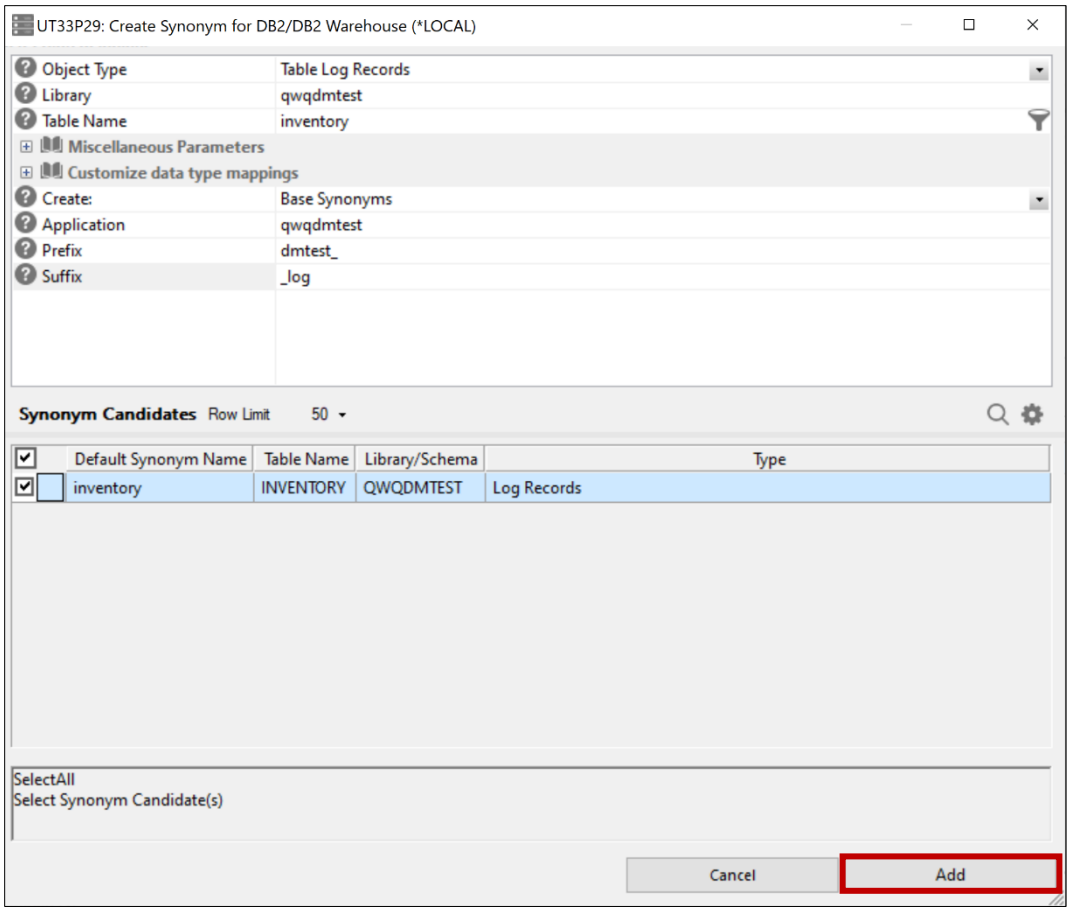


Figure 13 Creating the table log synonym

Let us look at the synonym we just created. Open the `dmtest_inventory_log` synonym. The `Properties` should open by default. Expand the `Variables` folder. This view shows all the columns, including the automatically added ones at the beginning, plus the global variables used to set the options described earlier.

Selecting any column shows its properties. Examine the contents of the synonym to become familiar with it.

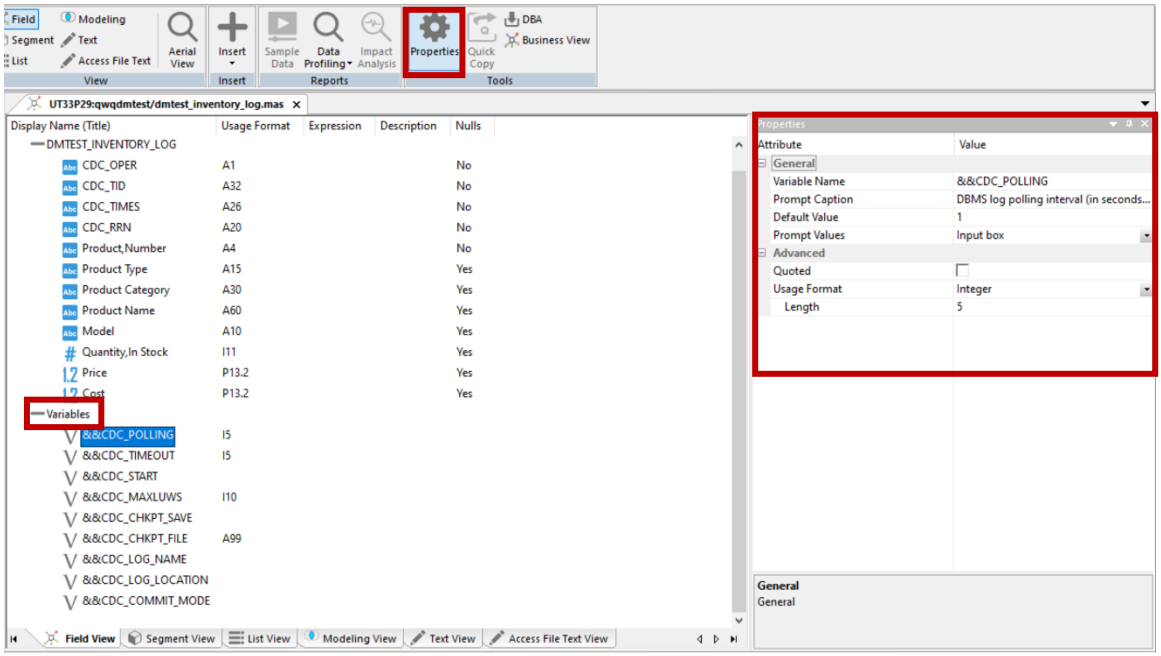


Figure 14 Viewing a table log record synonym

Now sample the data to verify the journal is being read. Right click on the synonym and click `Sample Data`. On the sample data window check all variables. Then click `Sample Data`.

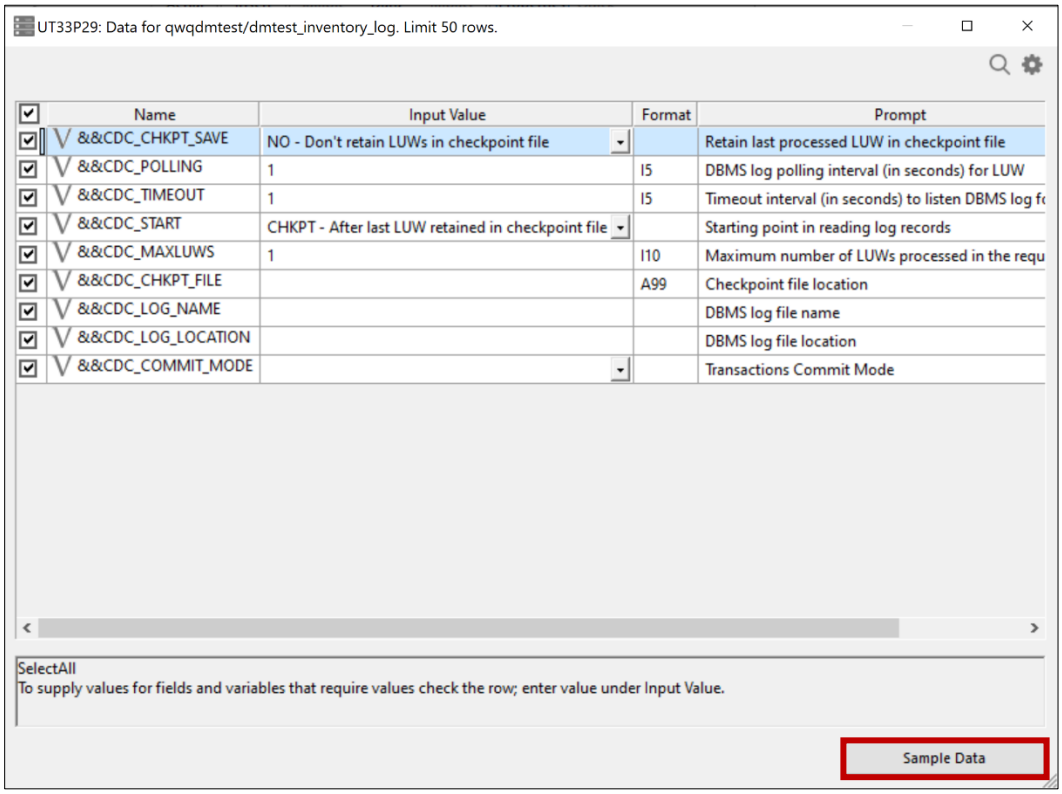
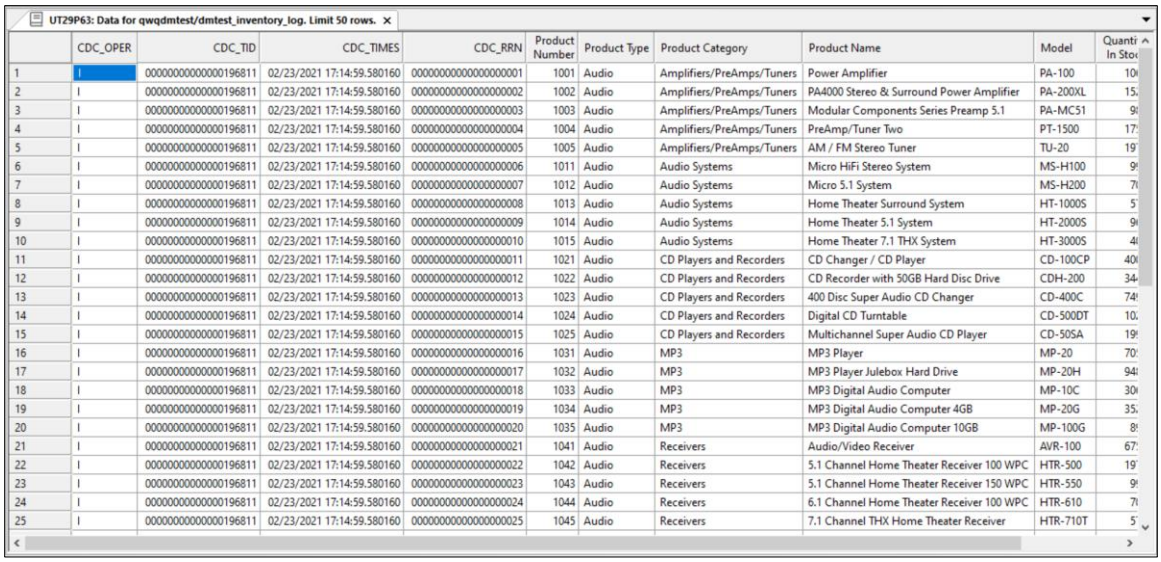


Figure 15 Sample Data dialog



UT29P63: Data for qwqdmtest/dmtest_inventory_log. Limit 50 rows.

	CDC_OPER	CDC_TID	CDC_TIMES	CDC_RRN	Product Number	Product Type	Product Category	Product Name	Model	Quanti In Sto
1	I	00000000000000196811	02/23/2021 17:14:59.580160	00000000000000000001	1001	Audio	Amplifiers/PreAmps/Tuners	Power Amplifier	PA-100	10
2	I	00000000000000196811	02/23/2021 17:14:59.580160	00000000000000000002	1002	Audio	Amplifiers/PreAmps/Tuners	PA4000 Stereo & Surround Power Amplifier	PA-200XL	15
3	I	00000000000000196811	02/23/2021 17:14:59.580160	00000000000000000003	1003	Audio	Amplifiers/PreAmps/Tuners	Modular Components Series Preamp 5.1	PA-MC51	9
4	I	00000000000000196811	02/23/2021 17:14:59.580160	00000000000000000004	1004	Audio	Amplifiers/PreAmps/Tuners	PreAmp/Tuner Two	PT-1500	17
5	I	00000000000000196811	02/23/2021 17:14:59.580160	00000000000000000005	1005	Audio	Amplifiers/PreAmps/Tuners	AM / FM Stereo Tuner	TU-20	19
6	I	00000000000000196811	02/23/2021 17:14:59.580160	00000000000000000006	1011	Audio	Audio Systems	Micro HiFi Stereo System	MS-H100	9
7	I	00000000000000196811	02/23/2021 17:14:59.580160	00000000000000000007	1012	Audio	Audio Systems	Micro 5.1 System	MS-H200	7
8	I	00000000000000196811	02/23/2021 17:14:59.580160	00000000000000000008	1013	Audio	Audio Systems	Home Theater Surround System	HT-1000S	5
9	I	00000000000000196811	02/23/2021 17:14:59.580160	00000000000000000009	1014	Audio	Audio Systems	Home Theater 5.1 System	HT-2000S	9
10	I	00000000000000196811	02/23/2021 17:14:59.580160	00000000000000000010	1015	Audio	Audio Systems	Home Theater 7.1 THX System	HT-3000S	4
11	I	00000000000000196811	02/23/2021 17:14:59.580160	00000000000000000011	1021	Audio	CD Players and Recorders	CD Changer / CD Player	CD-100CP	40
12	I	00000000000000196811	02/23/2021 17:14:59.580160	00000000000000000012	1022	Audio	CD Players and Recorders	CD Recorder with 50GB Hard Disc Drive	CDH-200	34
13	I	00000000000000196811	02/23/2021 17:14:59.580160	00000000000000000013	1023	Audio	CD Players and Recorders	400 Disc Super Audio CD Changer	CD-400C	74
14	I	00000000000000196811	02/23/2021 17:14:59.580160	00000000000000000014	1024	Audio	CD Players and Recorders	Digital CD Turntable	CD-500DT	10
15	I	00000000000000196811	02/23/2021 17:14:59.580160	00000000000000000015	1025	Audio	CD Players and Recorders	Multichannel Super Audio CD Player	CD-50SA	19
16	I	00000000000000196811	02/23/2021 17:14:59.580160	00000000000000000016	1031	Audio	MP3	MP3 Player	MP-20	70
17	I	00000000000000196811	02/23/2021 17:14:59.580160	00000000000000000017	1032	Audio	MP3	MP3 Player Julebox Hard Drive	MP-20H	94
18	I	00000000000000196811	02/23/2021 17:14:59.580160	00000000000000000018	1033	Audio	MP3	MP3 Digital Audio Computer	MP-10C	30
19	I	00000000000000196811	02/23/2021 17:14:59.580160	00000000000000000019	1034	Audio	MP3	MP3 Digital Audio Computer 4GB	MP-20G	35
20	I	00000000000000196811	02/23/2021 17:14:59.580160	00000000000000000020	1035	Audio	MP3	MP3 Digital Audio Computer 10GB	MP-100G	8
21	I	00000000000000196811	02/23/2021 17:14:59.580160	00000000000000000021	1041	Audio	Receivers	Audio/Video Receiver	AVR-100	67
22	I	00000000000000196811	02/23/2021 17:14:59.580160	00000000000000000022	1042	Audio	Receivers	5.1 Channel Home Theater Receiver 100 WPC	HTR-500	19
23	I	00000000000000196811	02/23/2021 17:14:59.580160	00000000000000000023	1043	Audio	Receivers	5.1 Channel Home Theater Receiver 150 WPC	HTR-550	9
24	I	00000000000000196811	02/23/2021 17:14:59.580160	00000000000000000024	1044	Audio	Receivers	6.1 Channel Home Theater Receiver 100 WPC	HTR-610	7
25	I	00000000000000196811	02/23/2021 17:14:59.580160	00000000000000000025	1045	Audio	Receivers	7.1 Channel THX Home Theater Receiver	HTR-710T	5

Figure 16 Sample data from the journal

You should see a sampling of the rows that were put into the journal.

> **NOTE**: If you do not see any sample data, it is likely that you either forgot to run the `CREATE TABLE` under commit or forgot to issue the subsequent `COMMIT`.

2.3.2 Creating Flows

With our synonyms created, it is time to create our flows. For simplicity we will create the 'target' tables in the same `qwqdmtest` library.

> **NOTE**: While putting the target tables into the same library as the source tables is possible to do (and we do it here) your target tables should be on another system from the source tables, or at least in another library on the same system.

To use a change data capture flow, there needs to be a base set of data in the target table. This reflects a common case for change data capture scenarios. At a minimum, two flows are used. The first flow is the 'bulk load' flow that is run once to populate the target tables with data to a certain point in time (usually current to the point when the flow runs). The second flow is used to capture and apply any changes that occur in the source tables after the first flow has been run.

For our scenario we will be creating and maintaining an inventory history table. This table keeps track of changing inventory daily and will allow us to track how inventory fluctuates over time.

2.3.3 Creating a Bulk Load Flow

To build the first 'bulk load' flow, right click on the `qwqdmtest` folder and click `New`, then `Flow`. Drag in `dmtest_inventory` from the left navigation tree onto the left side of the flow palette. Next, click and drag `New Target` from the ribbon onto the right side of the palette.

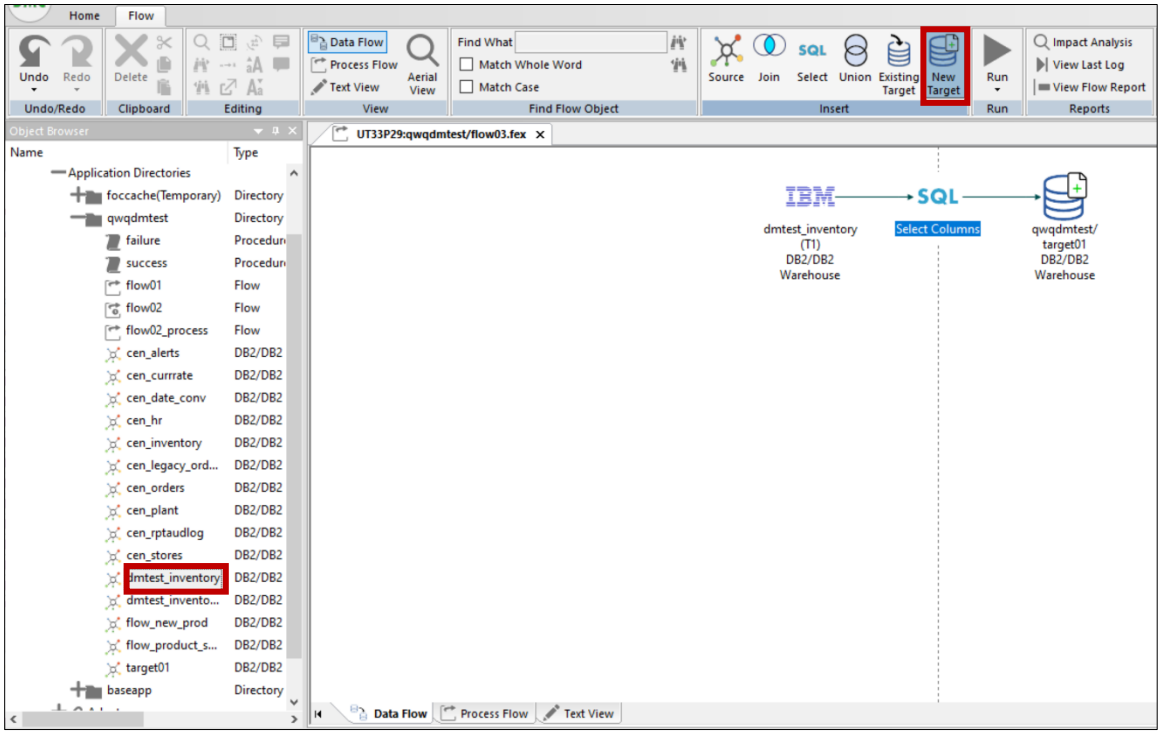


Figure 17 Creating a new flow

> **NOTE**: If you do not see `New Target` in the ribbon, widen your DMC window. The ribbon changes based on the window width and `New Target` could be 'hidden' because your DMC window is not wide enough to show it.

Right click on the `SQL` icon in the middle of the palette, select `Column Selection`. Highlight all columns and click the >> arrow to select all columns.

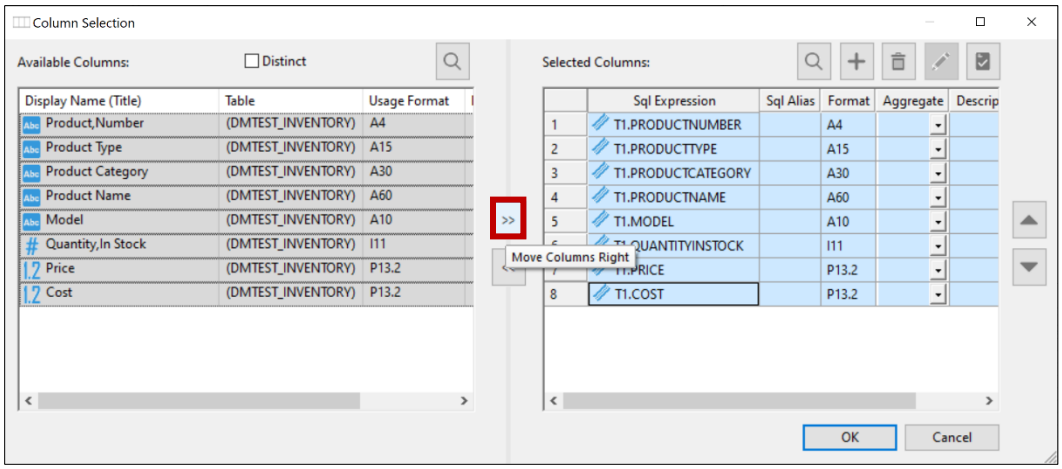


Figure 18 Selecting all of the columns

Since we are creating an inventory history, we need to add a date column. Click the `Insert Columns` button in the upper right to add the new column. Name the new column `REFDATE`. In the Expression box, click the `Date` button and select `Current Date`. Then click `OK`.

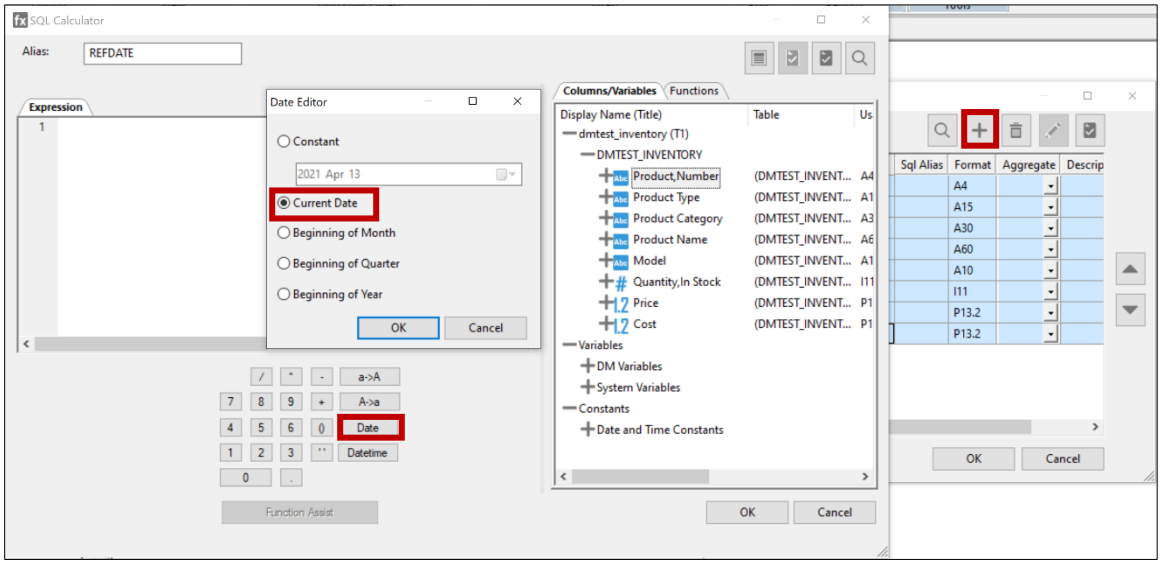


Figure 19 Inserting a new column

Move the new `REFDATE` column to the top of the Select Columns list by highlighting it and clicking the `up arrow` key. Then click `OK`.

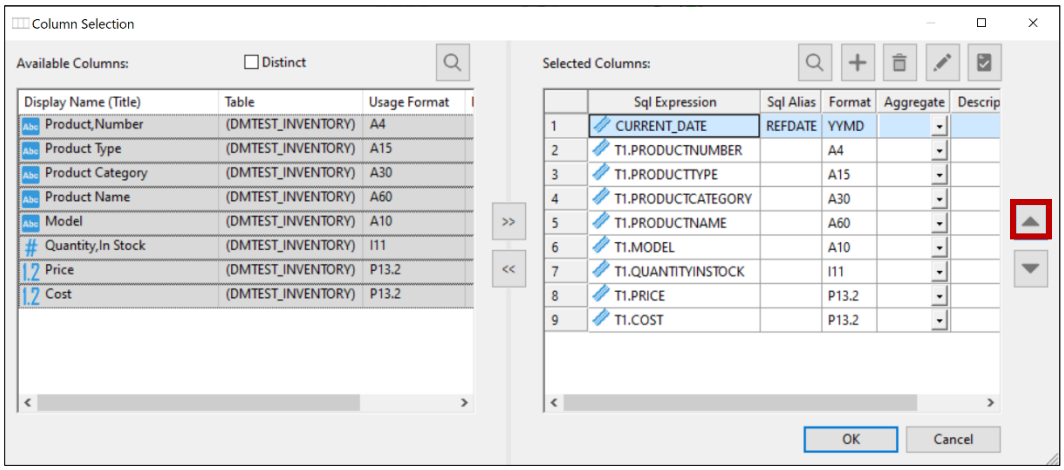


Figure 20 Moving REFDATE to the top of the selected columns list

Now we need to identify the target table. Right click on the target table icon and select `Properties`. Click on the ellipsis (…) to the right of the `KEY` entry. Select `REFDATE` and `PRODUCTNUMBER` and click `OK`. For `Table` and `Synonym`, type `qwqdmtest/inventory_history`. For `Load Type`, choose `Insert Records from Memory`.

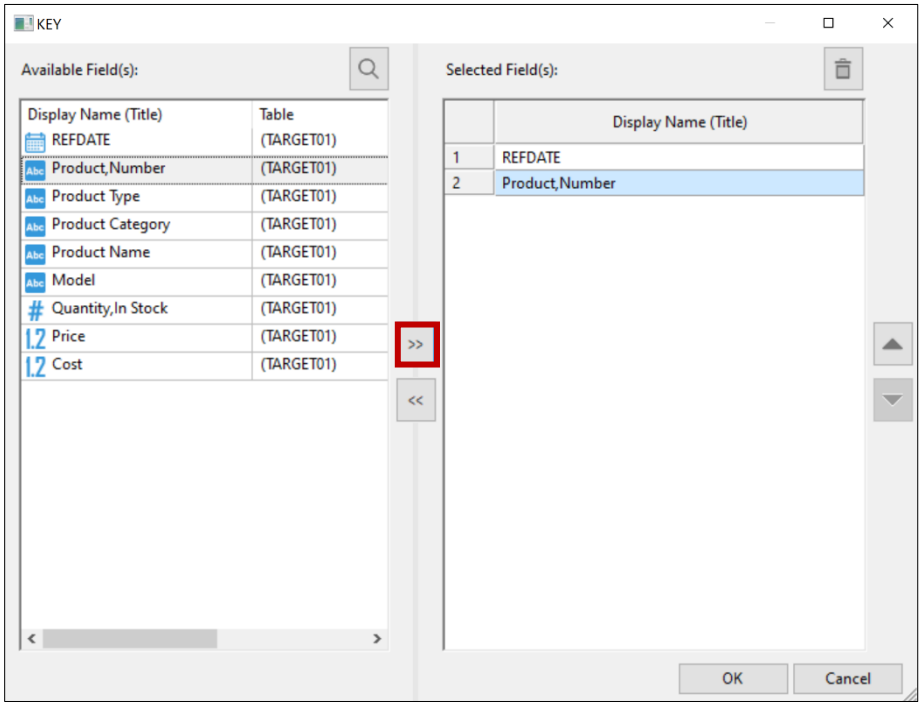


Figure 21 Selecting keys

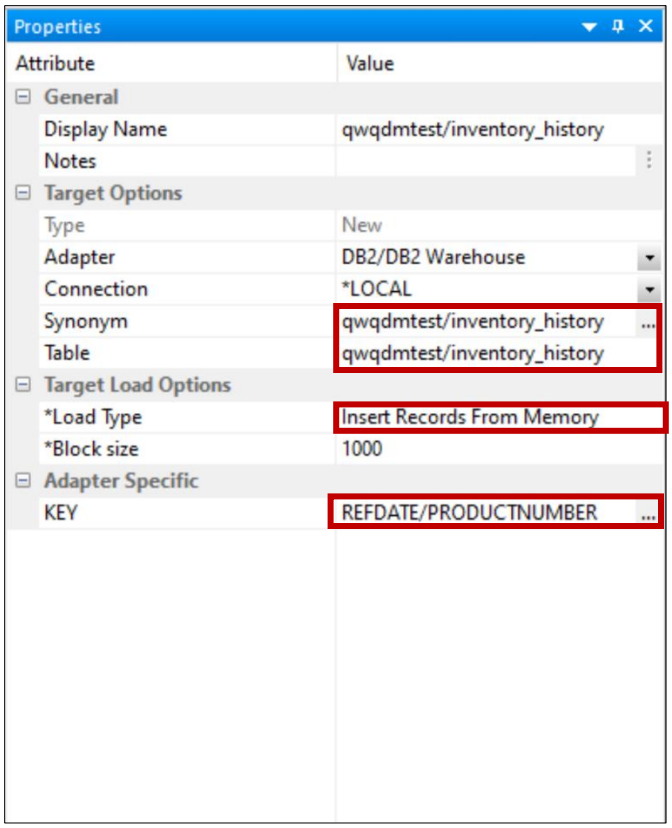


Figure 22 Changing the data target properties

Click `Save As` and name the flow `flow_bulk_load`**.** Then click `Save`.

On the left navigation tree find `flow_bulk_load` then right click on it and click `Run`. This will create the target table, `inventory_history`, and populate it with all the records from the data source. In the console log we can see the processing that took place when the flow was run.

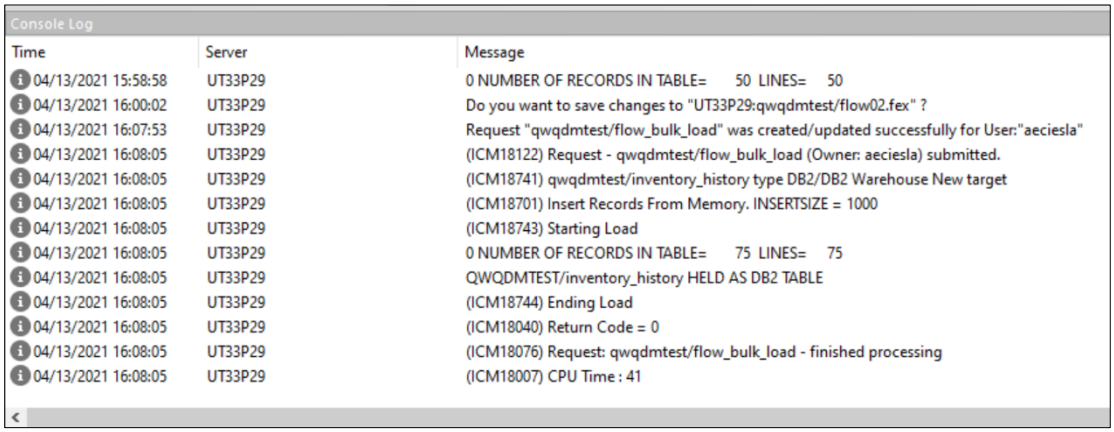


Figure 23 Console log output

Refresh the left navigation tree by right clicking on the server name (your system name) near the top of the tree and clicking `Refresh`. Under the `qwqdmtest` folder you should see the synonym `inventory_history`. Right click the synonym and click `Sample Data`.

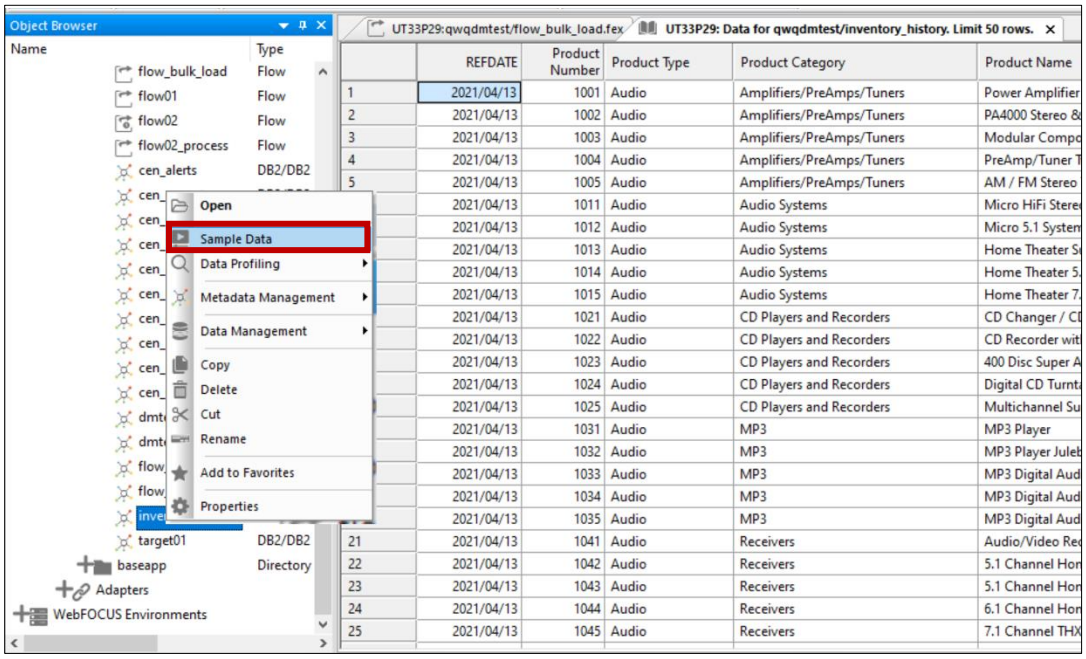


Figure 24 Sampling data from the new synonym

Close the sample data tab and the `flow_bulk_load` tab on the palette.

2.3.4 Creating a CDC Maintenance Flow

Now we are ready to create the second flow. This flow will capture any changes in the data source table and apply them to the target table `inventory_history`. We want `inventory_history` to contain a history of our inventory as it changes over time. That way we can go back and see how our inventory fluctuates. We will only capture inventory when it changes.

To build the flow, right click on the `qwqdmtest` folder and click `New`, then `Direct Load Flow`. A Direct Load Flow implies the data will come in and be processed right away. Drag in `dmtest_inventory_log` from the left navigation tree onto the left side of the flow palette. Click on `Existing Target` from the ribbon and click `inventory_history`. Then click `Select`.

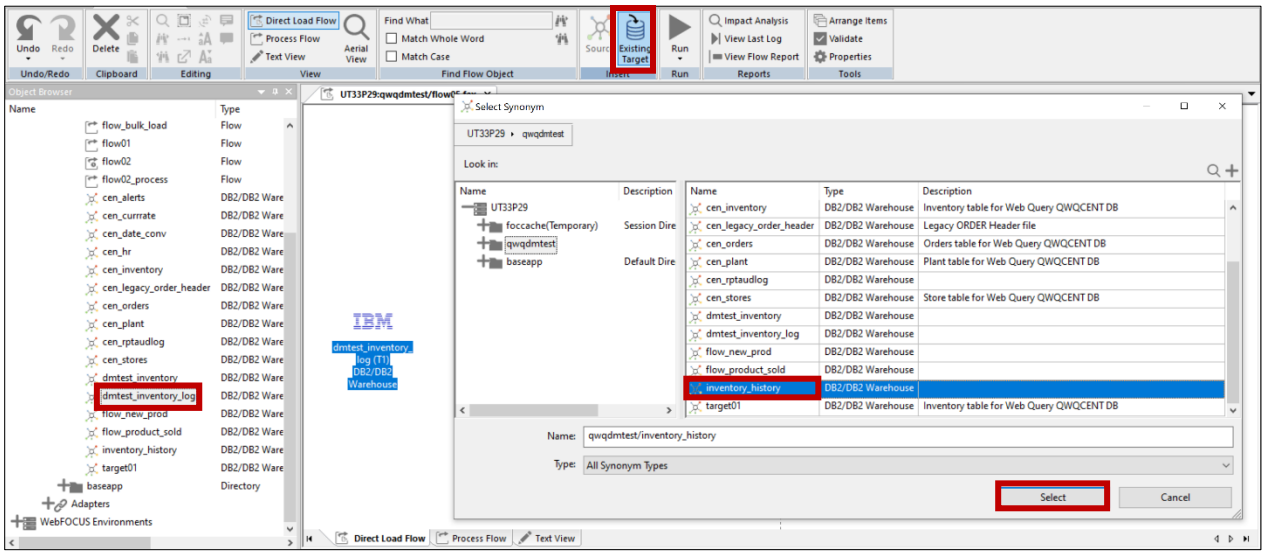


Figure 25 Creating a new direct load flow

The `inventory_history` synonym should show up on the right of the palette and be connected to the `dmtest_inventory_log` synonym. Right click on `inventory_history` on the palette and click `Target Transformations`. In the Transformations window, click the automap button at the top to have all matching columns automatically selected.

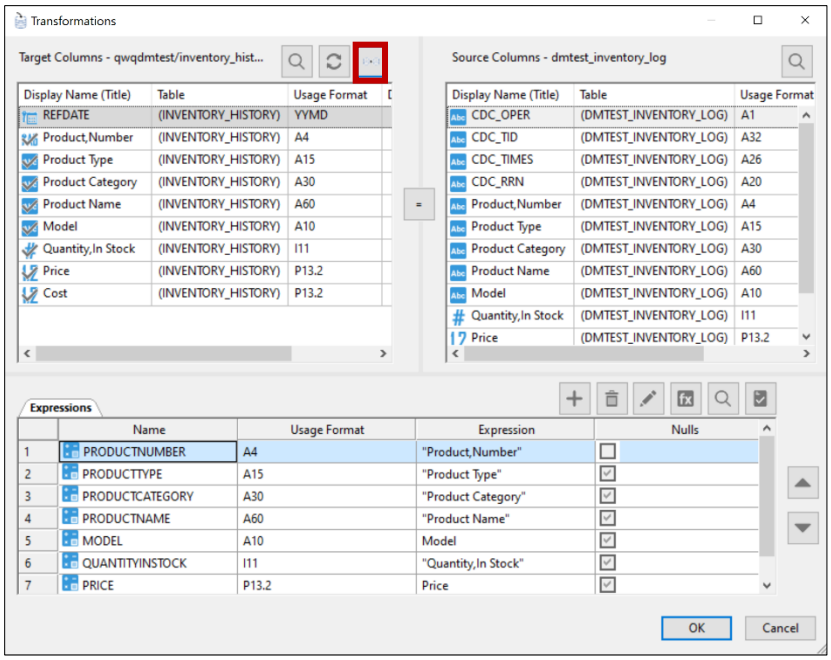


Figure 26 Automapping the columns

Now we want to add a current date field like we did for bulk load. Click the `Insert Intermediate Transforms` button. In the Transformation Calculator, name the field `REFDATE`. Then click the `Date` button and choose `Current Date`. In the Format field, specify `YYMD`.

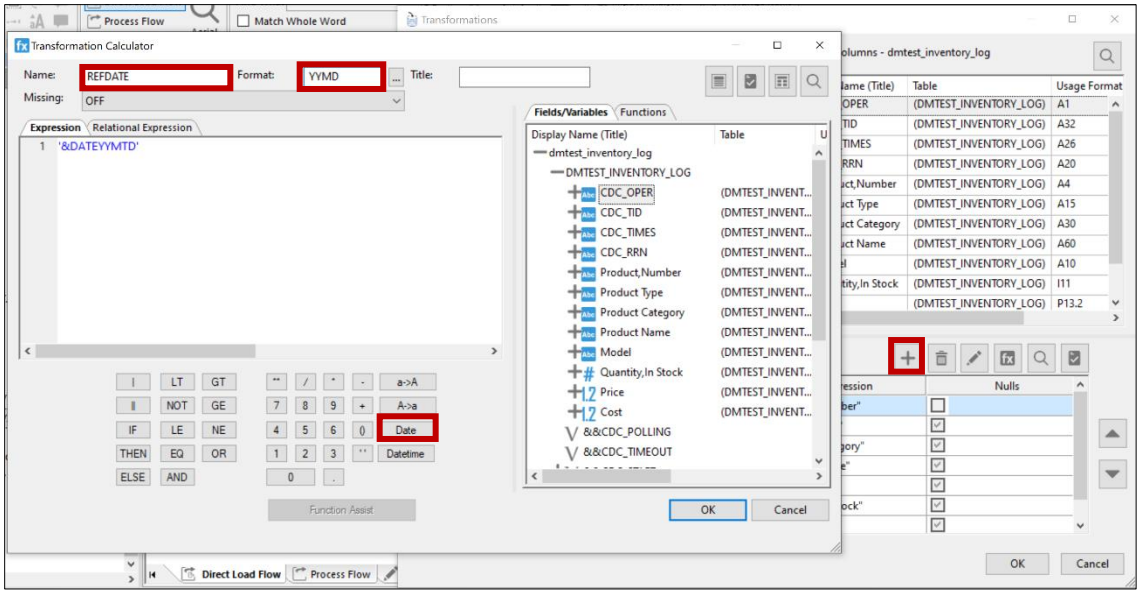


Figure 27 Inserting a transform for current date

Click OK to create the new REFDATE column. Now move REFDATE to the top of the column list by highlighting it and clicking the up arrow key. Then click OK.

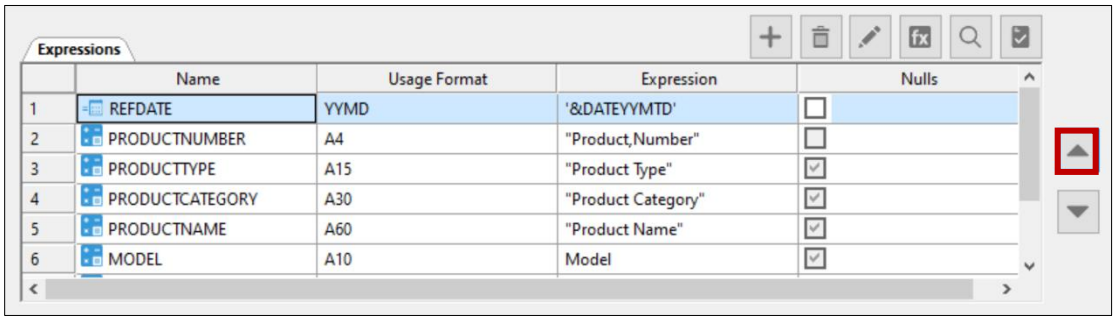


Figure 28 Moving REFDATE to the top of the list

This flow will be capturing changes to the inventory table (via the journal) and putting it into the inventory_history table. We will be capturing the change once a day. However, for any given day it is possible that there could be multiple changes to a particular product in the inventory. Which of these changes do we capture in the history table for that day?

Since we will only keep one inventory change per day per product, we will keep the last change for a given day. That means two things:

- When a row is read from the journal and we see there is already a row in the inventory_history table for the given product on the current date, we will replace that row in the history table with the incoming row. This means the last product inventory change for the current date will ultimately be the one we retain in the history table.
- When a row is read from the journal and there is not a row in the history table for the product for the current date, then it is added to the history table.

We prepared for this by defining the Key for the history table to be REFDATE and PRODUCTNUMBER. This key defines what it means to have a 'matching' row in the history table. Therefore, anytime a new row coming in from the journal has the same REFDATE and PRODUCTNUMBER value as an existing row in the inventory_history table, we have a match and must decide what to do. In this case we want to replace the row in the history table with the incoming row.

Back on the palette, right click on the inventory_history icon and click Properties. Set Load Type to Insert/Update. For If the record exists, choose Update the existing record. This will cause the incoming record to be used and the existing record in the history table will be overwritten. For If the record does not exist, choose Include the record. This will cause the incoming record to be added to the history table if there is not an entry for the product on the current date.

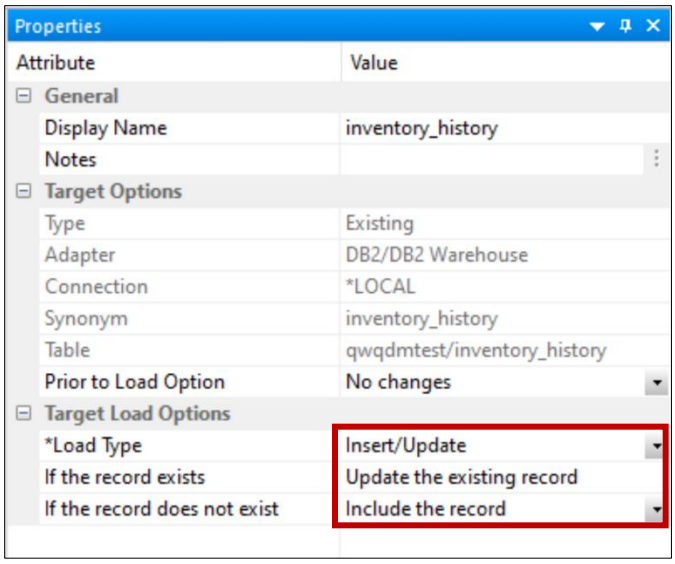


Figure 29 Setting the properties for inventory_history

Save this flow using `Save As` and name it `flow_delta_changes`.

Now we will run this Direct Load Flow. Find `flow_delta_changes` in the left navigation tree and right click on it, then click `Run`. Look at the console log at the bottom of DMC.

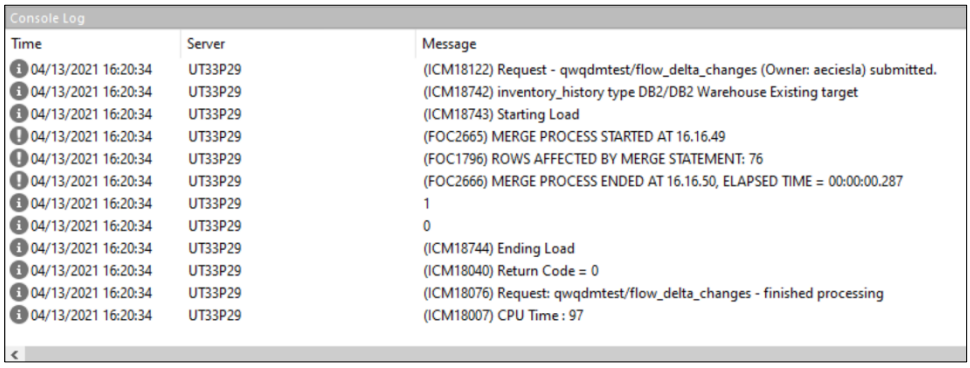


Figure 30 Console log output

Notice that 76 rows were processed by the merge statement. In other words, it reprocessed the 75 rows that the bulk load already processed. This is because the journal contains all the entries in inventory since we created/populated the table under commit against this journal. If this had been case where inventory was an older file with records existing prior to journaling being started, the bulk load would capture all the base records, thereby providing the base records for the subsequent direct flow runs.

Let us run the Direct Load Flow a second time. Right click on `flow_delta_changes` in the tree and click `Run`. Look at the console log at the bottom of DMC.

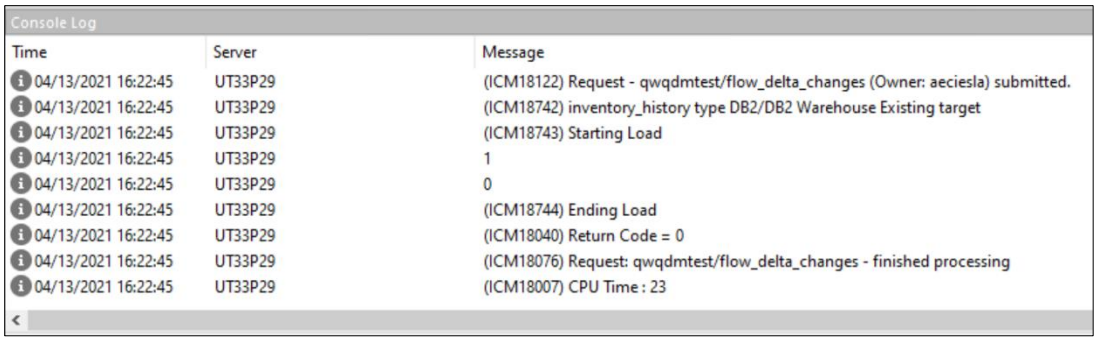


Figure 31 Console log output

On this second run, no rows were processed. This shows an important aspect of table log synonyms called a *log checkpoint file*. Each time a flow involving a table log synonym runs, a checkpoint file (a file in IFS) is updated with information about the last journal entry read. When the flow is run again, it starts reading in the journal after the last position recorded in the checkpoint file. If subsequent journal entries are added, the flow will pick up those new journal entries, apply them and update the checkpoint file to indicate the new ending position. So, we can run `flow_delta_changes` repeatedly knowing it will only process new entries.

Now that we have seen the checkpoint process in action, we will take a closer look at it. In the left navigation tree, right click on the table log synonym `dmtest_inventory_log` and click `Sample Data`. Click the check box for all the `&&CDC_` variables and click `Sample Data`.

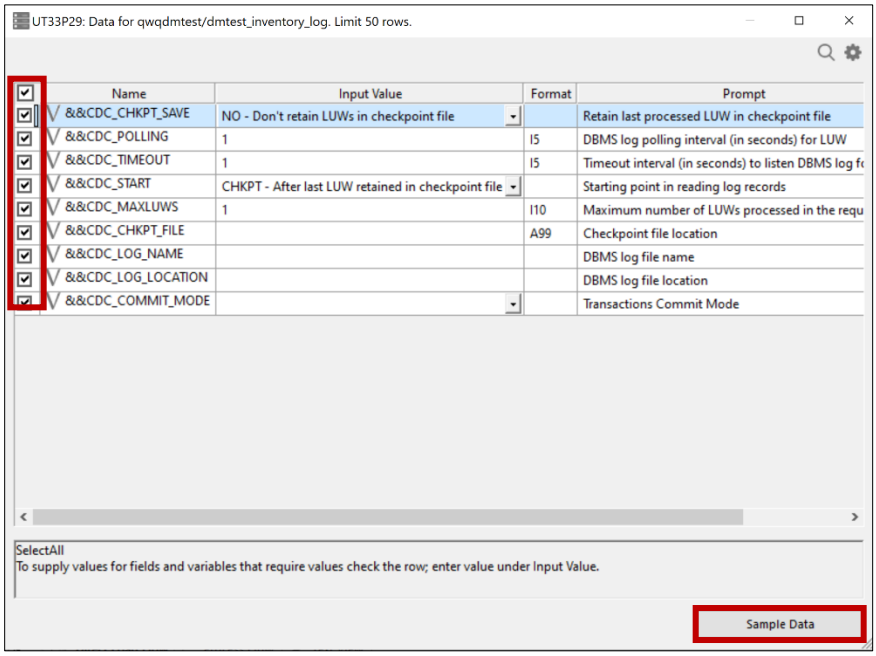


Figure 32 Including all &&CDC_ variables

Notice that no records were displayed. This is because the checkpoint file has saved the last read position, which was the last entry in the journal.

Let us sample data again with one difference. In the left navigation tree, right click on the table log synonym `dmtest_inventory_log` again and click `Sample Data`. Click the check box for all the `&&CDC_` variables as before. But this time click on the dropdown box for `&&CDC_START` and choose `CUR_LOG – First available LUW in DBMS log`. Then click `Sample Data`.

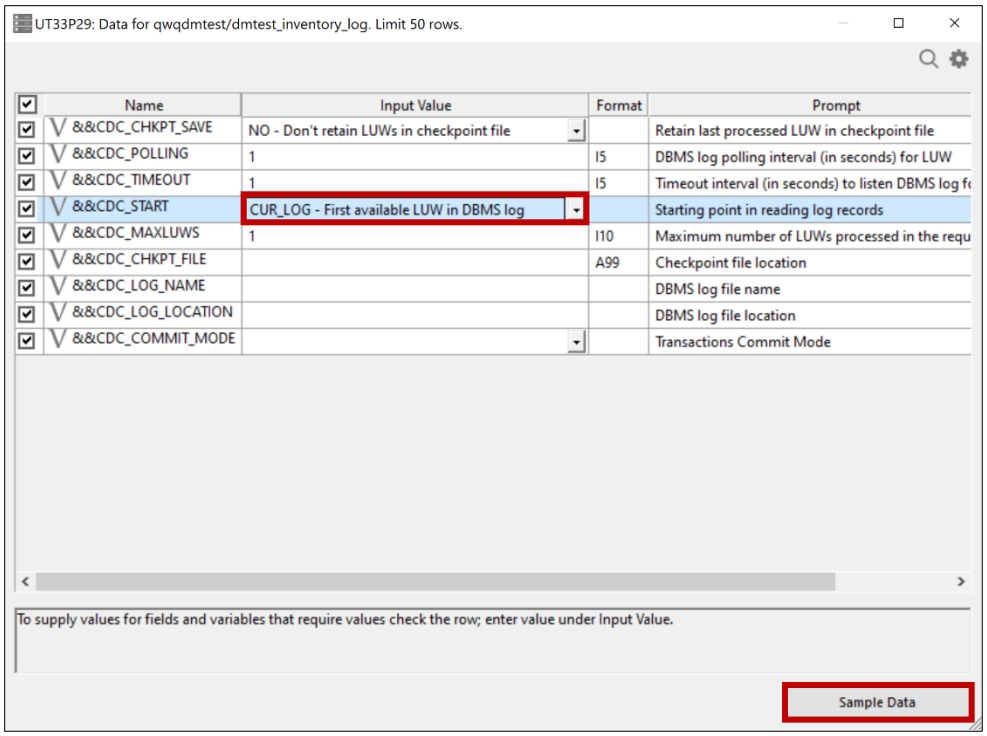


Figure 33 Changing the start position of the log (journal)

This time we got a full sample of records from the log (journal). This shows a very useful feature for CDC processing. It is possible to control whether to maintain position in the journal as entries are processed (the default), restart from the beginning of the journal, or start from a point in time in the journal that corresponds to when the flow job started.

> **NOTE**: The change of value for the `&&CDC_START` variable was temporary; it did not change the value permanently for the synonym. To do a permanent change, we would open the synonym, select the variable, change its default, then save the synonym.

Two other variables that work with &&CDC_START are &&CDC_CHKPT_SAVE and &&CDC_CHKPT_FILE.

- &&CDC_CHKPT_SAVE defines whether the checkpoint file is updated at the end of the journal read processing in the flow. Normally this is set to YES to indicate the position should be saved (updated). However, in test scenarios it can be useful not to save position so that repeatedly running the flow will process the same entries over and over.
- &&CDC_CHKPT_FILE defines the (IFS) file that will retain the checkpoint information. By default the directory and name is /qibm/userdata/qwebqry/apps/*<application directory>/<synonymname>*.chp. However, you can change this to a different name if you wish. This is sometimes useful if you want to maintain different checkpoint positions for different flows.

Now we will see the delta flow in action. First, we need to make some row changes in the data source so that new journal entries are created. Through an SQL interface, update a few rows in the source table with the following SQL statements. This will reduce the inventory of all products within the productnumber range by 150.

> **NOTE**: Make sure to run under commitment control like we did before.

```
UPDATE QWQDMTEST.INVENTORY
SET QUANTITYINSTOCK = QUANTITYINSTOCK - 150
WHERE PRODUCTNUMBER BETWEEN '1000' and '1010';
COMMIT;
```

We could just run the delta flow and see the changes apply. However, remember we are using the current date as the value for CURDATE. Assuming you are running these flows on the same day, our inventory_history file will only have the current date as the value of the CURDATE column in all its rows. We would like to see our UPDATE above show as a history that is, over more than one day. Therefore, we will cheat and first update the existing rows of target table inventory_history to set the date to an older date. Run this SQL statement:

```
UPDATE QWQDMTEST.INVENTORY_HISTORY
SET REFDATE = REFDATE - 1 DAY;
COMMIT;
```

In effect we changed the history table to look like it was populated yesterday.

Now run the Direct Load Flow. Right click on `flow_delta_changes` in the tree and click `Run`. Look at the console log at the bottom of DMC.

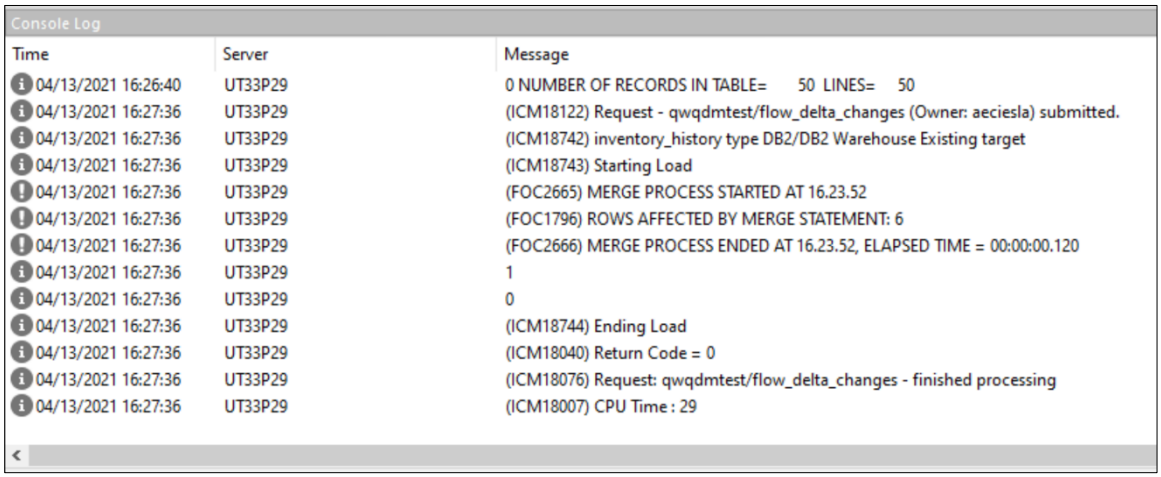


Figure 34 Console log output after updating the database

On this run, the source table rows we updated were processed. We can verify this by looking in the `inventory_history` table for those rows using the SQL statement:

```
SELECT REFDATE, PRODUCTNUMBER, QUANTITYINSTOCK
FROM QWQDMTEST.INVENTORY_HISTORY
WHERE PRODUCTNUMBER BETWEEN '1000' and '1010'
ORDER BY PRODUCTNUMBER
```

We now have two days of history for these products. Each day has a different `quantityinstock` that reflects that we reduced the amount by `150`. If we were planning to implement this flow in our environment, the next step would be to schedule `flow_delta_changes` to run every day to capture any changes from our source environment. However, since we covered scheduling in a prior chapter, we will not cover that here.

2.4 CDC for Remote Journals

DataMigrator allows you to perform change data capture on remote journals. To read a journal, DataMigrator *must* be running where the remote journal and journal receiver are. Using remote journaling allows DataMigrator to run on a separate system from the main system where the file resides. For example, it can run on a partition other than the production partition.

This concept is illustrated in the figure below.

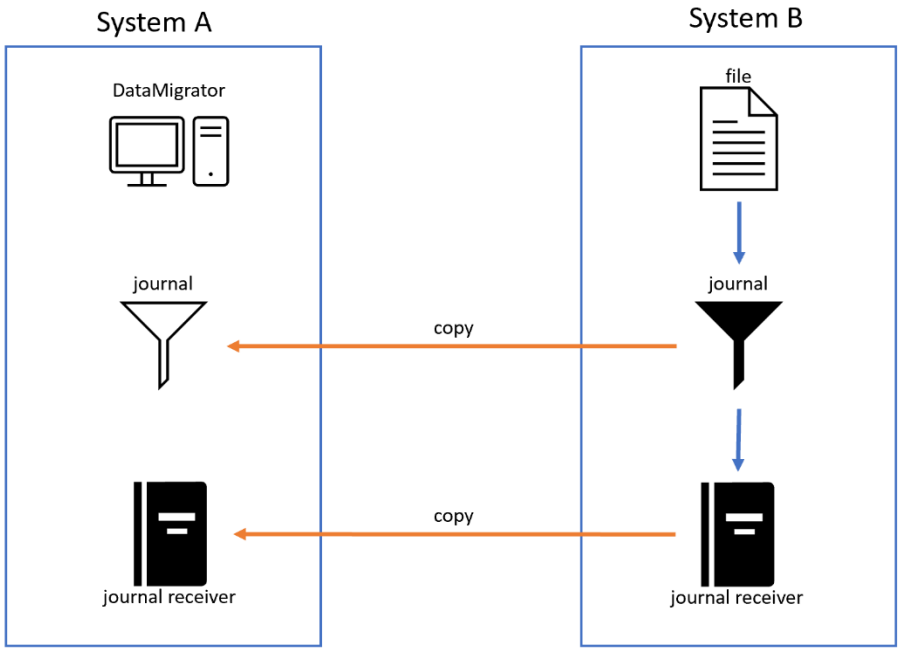


Figure 35 Remote journaling with DataMigrator

The figure above illustrates some key concepts for setting up CDC using remote journals. System A represents the remote system where DataMigrator is running and System B represents the source system that contains the file that is being journaled. The operating system creates the remote journal receiver on System A when the file on System B is set up for remote journaling. Then there must be an empty copy of the file and journal on System B in System A which allows DataMigrator to interpret the format of the journal entries.

2.4.1 Setup

First, we will need to set up a journal and journal receiver on the source system. Unlike in the previous section, we will use SQL to create a schema which will automatically create the journal called `QSQJRN` and any tables associated with it. Issue the following SQL statement:

```
CREATE SCHEMA DMTEST;
```

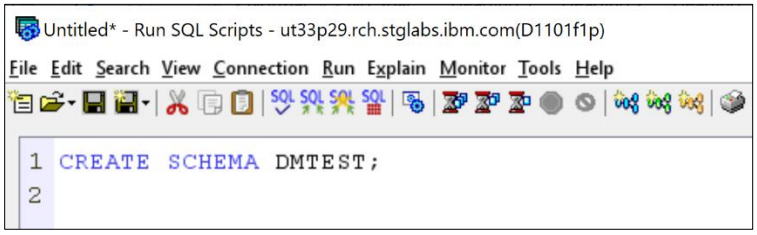


Figure 36 Creating a schema

Then we will populate a table in the schema with data. Issue the following SQL commands. Make sure to be running under some level of commitment control, for example `*CHG`.

> **NOTE**: To run under commitment control through Access Client Solutions (ACS) Run SQL Scripts, click `Connection`, then click `Connected - xxx` where `xxx` is your system and click `JDBC Settings`. Then change the `Isolation level`.

```
CREATE TABLE DMTEST.INVENTORY AS
(SELECT * FROM QWQCENT.INVENTORY) WITH DATA;
COMMIT;
```

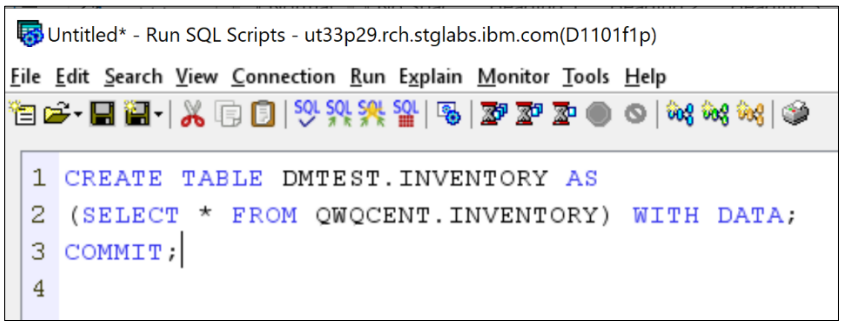


Figure 37 Creating a table for inventory data

Next, we need to setup the connection to the remote system. On the remote system, enter the CL command `WRKRDBDIRE`. This shows a list of the remote system connections. Take note of the entry corresponding to *LOCAL which gives you the local name of the remote system.

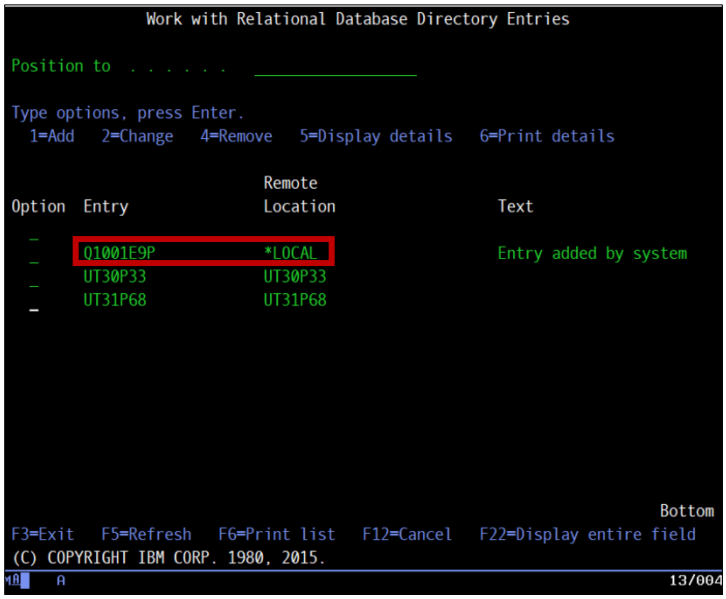


Figure 38 List of remote system connections

To add a remote connection, enter the CL command `ADDRDBDIRE` on the source system. For `Relational database`, enter the local name of the remote system. For `Remote Location`, enter the name of the remote system. For `Type`, use `*IP`.

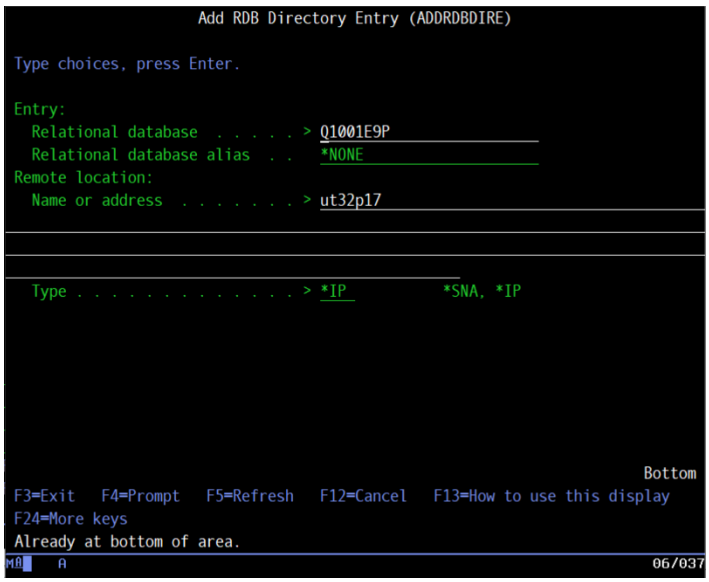


Figure 39 Adding a remote connection

Now we will add the remote journal and activate it. First, on the remote system enter the CL command `CRTLIB DMTESTER`. This creates an empty library we will use to hold the remote journal. On the source system, enter the CL command `ADDRMTJRN`. For `Relational database`, enter the local name of the remote system. For `Source journal`, enter `QSQJRN` in the library `DMTEST`. For target journal, enter `QSQJRN` in the library `DMTESTER`. For `Remote Receiver library`, enter `DMTESTER`. For `Remote Journal Type`, enter `*TYPE2` since the journal libraries on the systems have different names. Make sure that `Delete receivers` is set to `*NO`.

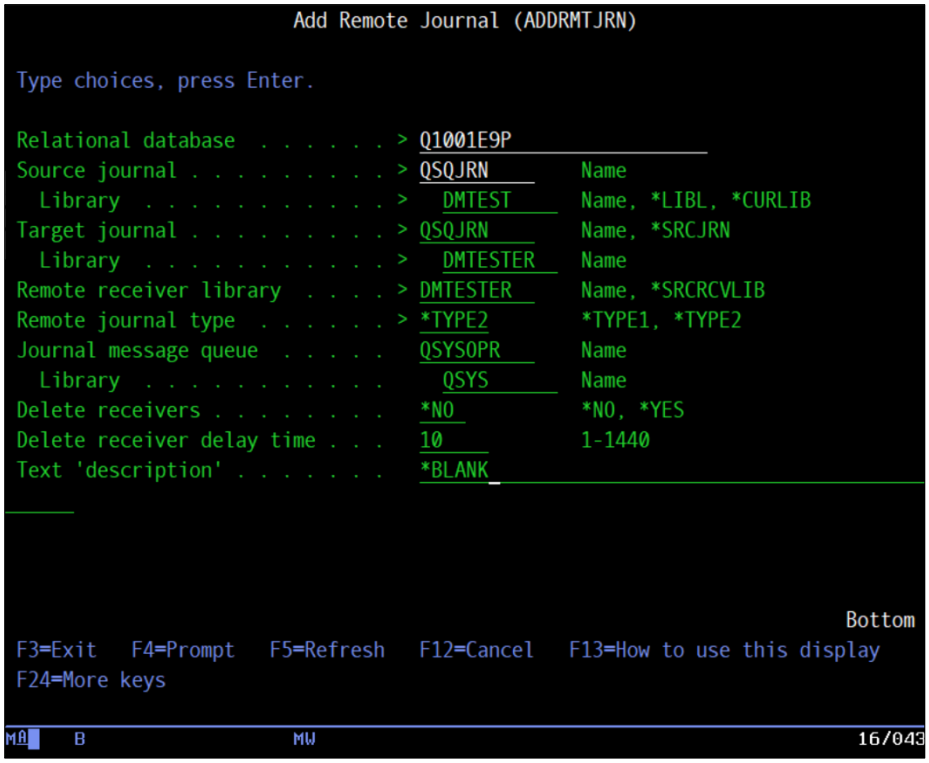


Figure 40 Adding a remote journal

You can verify the journal was added to the remote system using the CL command `WKRLIB DMTESTER`. You can see that the library only contains the journal, `QSQJRN`, and not the journal receivers. We must activate the journal to add the journal receivers.

Back on the source system, enter the CL command `CHGRMTJRN`. For `Relational database`, enter the local name of the remote system. For `Source journal`, enter `QSQJRN` in the library `DMTEST`. For target journal, enter `QSQJRN` in the library `DMTESTER`. For the `Journal state`, enter `*ACTIVE`. Hit `Enter`. In `Delivery Mode`, enter `*ASYNC`. For `Staring journal receiver`, you can choose to use `*ATTACHED` or the name of the first journal receiver on the source system. Using the name of the first journal receiver will transfer a full history of changes to the remote system.

NOTE: You can use `*SYNC` as the delivery mode, however it will decrease performance significantly since journal entries must be written to memory on both the source and remote systems.

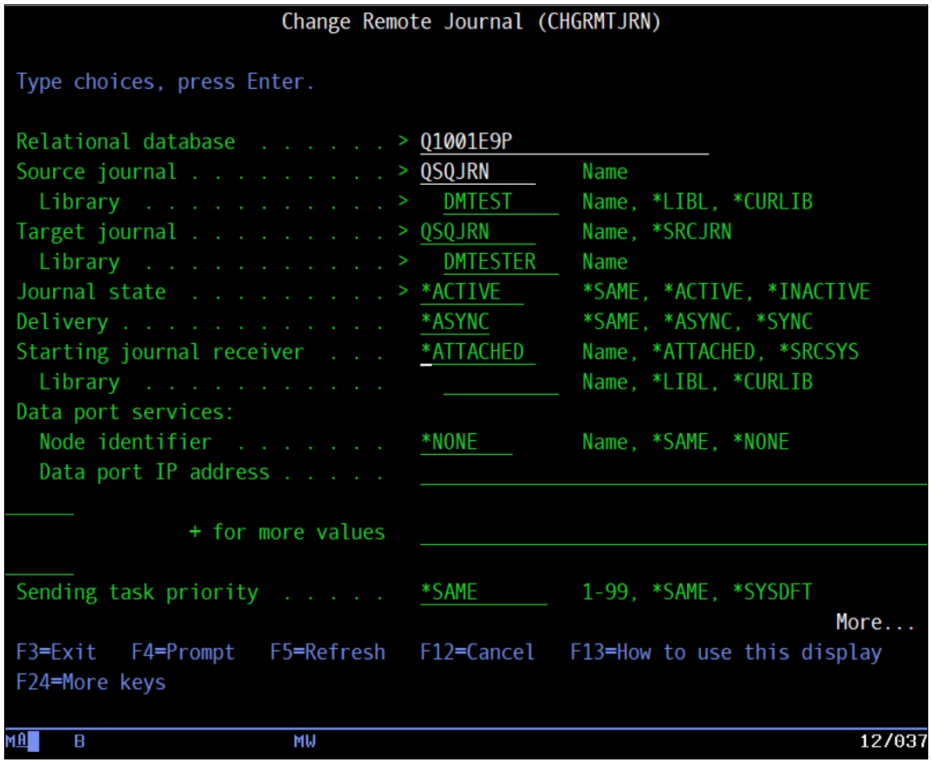


Figure 41 Activating a remote journal

Now if you revisit the library `DMTESTER` on the remote system using the CL command `WRKLIB DMTESTER`, you will see the journal `QSQJRN` and any journal receivers.

To use CDC with a remote journal, we need an empty copy of the source database file on the remote system with the same name and same library as the source table. This tells the remote system how to interpret the change data it receives over the network from the source system. On the source system, enter the CL command `CRTLIB DMTESTER`. Then enter the CL command `CRTDUPOBJ`. For `From Object`, enter `INVENTORY` in the library `DMTEST`. For `Object type`, enter `*FILE`. For `To library`, enter `DMTESTER`. Hit `Enter`. Make sure that `Duplicate data` is set to `*NO`.

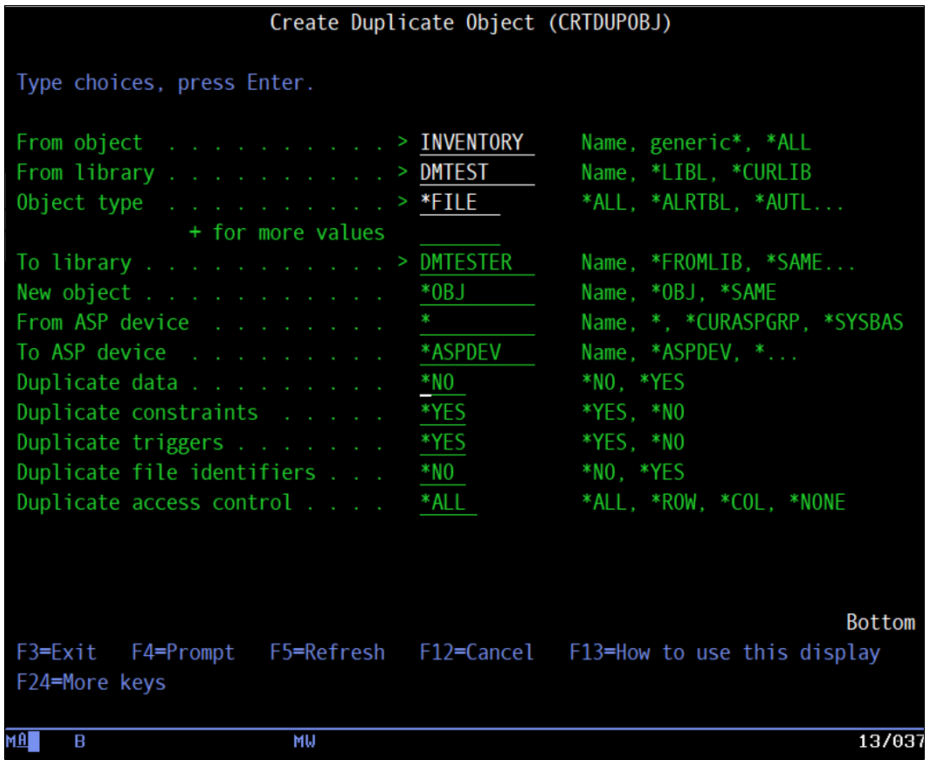


Figure 42 Creating an empty copy of the source table

We need to create an empty save file that will allow us to 'zip' the empty copy of the source table and send it to the remote system. Enter the CL command `CRTSAVF`. For `Save file`, enter any name such as `transfer`. For `Library`, enter `DMTESTER`.

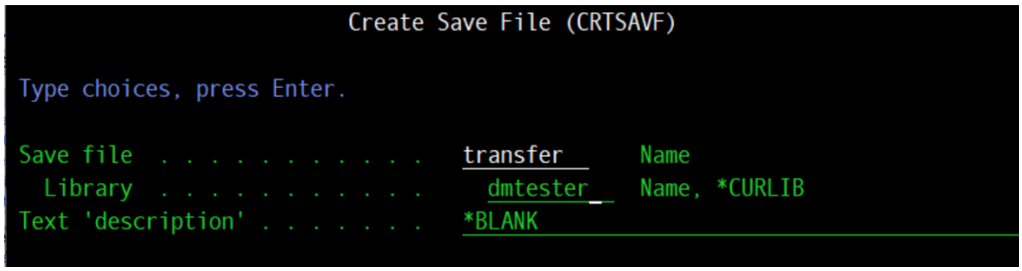


Figure 43 Creating an empty save file

We can 'zip' up the empty copy of the table. Enter the CL command `SAVOBJ`. For `Objects`, enter `INVENTORY`. For `Library`, enter `DMTESTER`. For `Device`, enter `*SAVF`. Hit `Enter`. For `Save file`, enter the name of the save file created above. In this case enter `transfer`. For `Library`, enter `DMTESTER`.

Figure 44 Creating the save file

On the remote system, we need to create an empty save file to 'unzip' the empty copy of the table to. Enter the CL command `CRTSAVF`. For `Save file`, enter the name of the save file, in this case `transfer`. For `Library`, enter `DMTESTER`.

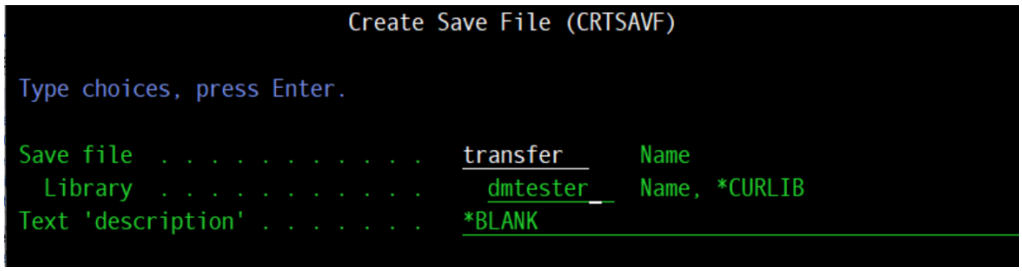


Figure 45 Creating an empty save file

Now we must move the save file from the source system to the remote system. Still on the remote system, enter the command `ftp <fully qualified domain name of source system>`. You will be prompted to log into the source system. Enter the command `bin`, to transfer in binary mode. Then enter the command `get dmtester/transfer dmtester/transfer (replace`. This will move the save file from the source system to the remote system.

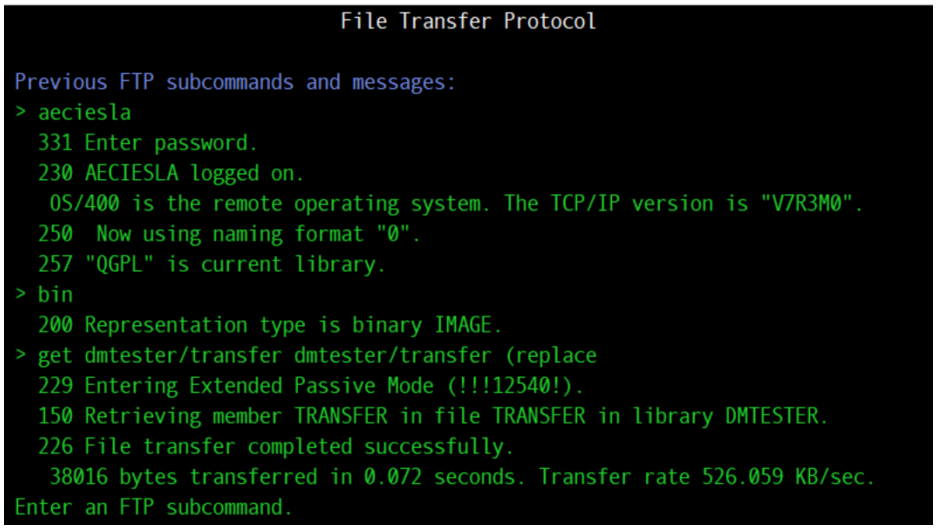


Figure 46 Moving the save file to the remote system

We need to restore the save file to 'unzip' its contents on the remote system. Back on the command line, enter the command `RSTOBJ`. For `Objects`, enter `*ALL`. For `Saved Library`, enter `DMTESTER`. For `Device`, enter `*SAVF`. Hit `Enter`. For `Save file`, enter then name of the save file, in this case `transfer` in library `DMTESTER`.

Figure 47 Restoring the save file on the remote system

When using a remote journal for CDC, DataMigrator requires an empty copy of the source table on the remote system with the same name and same library as the source table. However, the names and libraries of the journals do not need to be the same. We will journal the empty copy of the source file on the remote system, create a synonym over it, then configure the synonym to point to the remote journal.

On the remote system, issue the following SQL statement:

```
CREATE SCHEMA DMTEST;
```

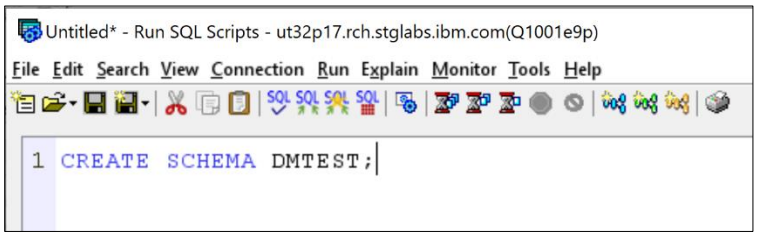


Figure 48 Creating a schema

Still on the remote system, enter the CL command `RSTOBJ`. For `Objects`, enter `*ALL`. For `Saved library`, enter `DMTESTER`. For `Device` enter `*SAVF`. Hit `Enter`. For `Save file`, enter `TRANSFER` in the library `DMTESTER`. Hit `F10`. Page down to find `Restore to library` and enter `DMTEST`.

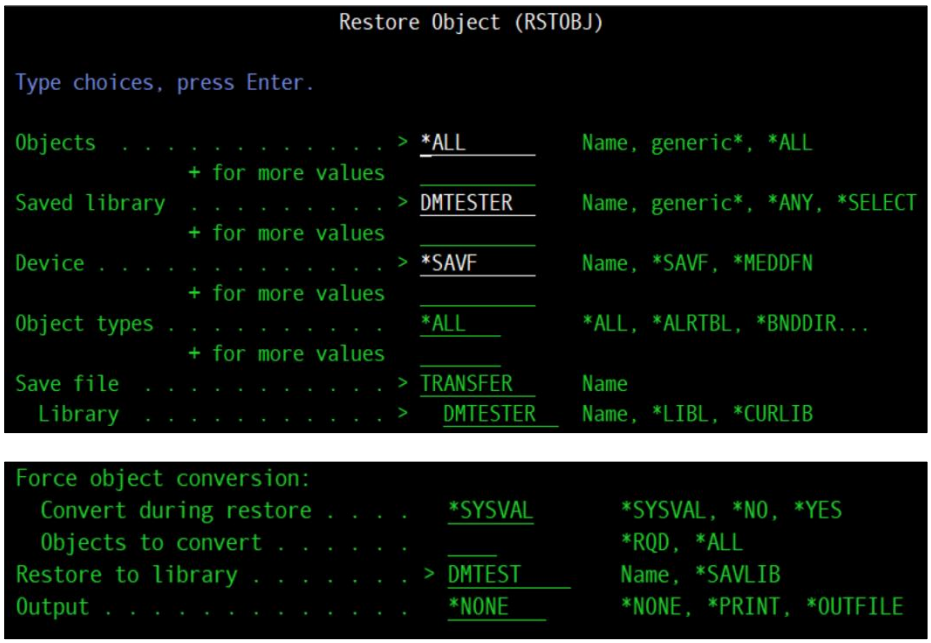


Figure 49 Restoring the inventory table

Enter the CL command `STRJRNPF`. For `Physical file to be journaled`, enter `INVENTORY` in library `DMTEST`. For `Journal`, enter `QSQJRN` in library `DMTEST`.

```
                  Start Journal Physical File (STRJRNPF)

Type choices, press Enter.

Physical file to be journaled  .   INVENTORY     Name, generic*, *ALL
  Library  . . . . . . . . . . .     DMTEST       Name, *LIBL, *CURLIB
             + for more values
                                     *LIBL
Journal  . . . . . . . . . . . .   QSQJRN        Name
  Library  . . . . . . . . . . .     DMTEST       Name, *LIBL, *CURLIB
Record images  . . . . . . . . .   *AFTER        *AFTER, *BOTH
Journal entries to be omitted  .   *NONE         *NONE, *OPNCLO
Logging level  . . . . . . . . .   *ERRORS       *ERRORS, *ALL
```

Figure 50 Journaling the inventory table

Now open the DMC and connect to the server where DataMigrator is running. We need to set the application path to add the test library to the DMC. Right click on `Application Directories` under the server name and click `Manage`, then `Application Path`.

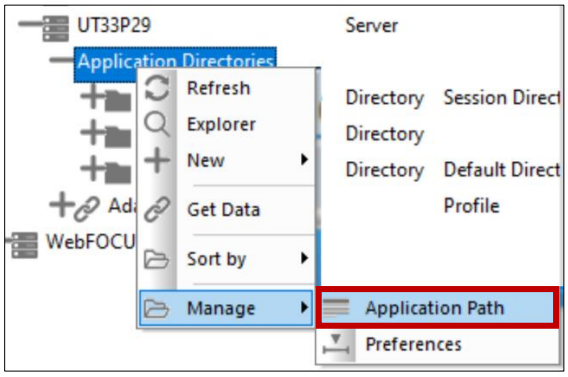


Figure 51 Accessing the application path

This opens the Application Path Configuration window. Click `Change Profile Precedence` and select `Append to previously executed profiles`.

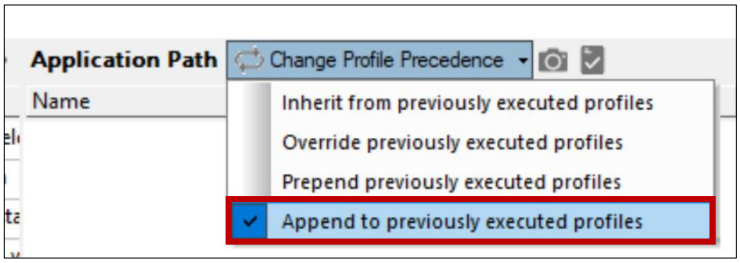


Figure 52 Changing the profile precedence

Find the test folder created earlier in guide, right click it and click `Add to Path`. The folder will show up on in the righthand column. Then click `Save` and `OK`.

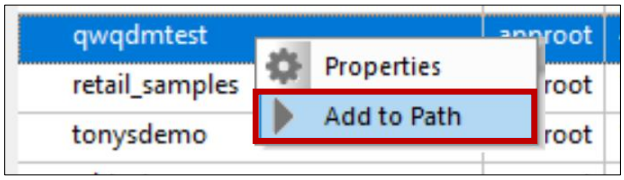


Figure 53 Adding the test folder to the application path

Next, we need to create a synonym over the remote journal. Right click on your test folder and click `New`, then `Synonym`. Choose `*LOCAL`. For `Object type`, choose `Table Log Records`. For `Library`, enter `DMTEST`. Then click `Search`. Click the checkbox next to `inventory`. For `Prefix`, enter `DMTEST_` and for `Suffix` enter `_jrn`. This will help you to identify the synonym later. Then click `Add` and close the window.

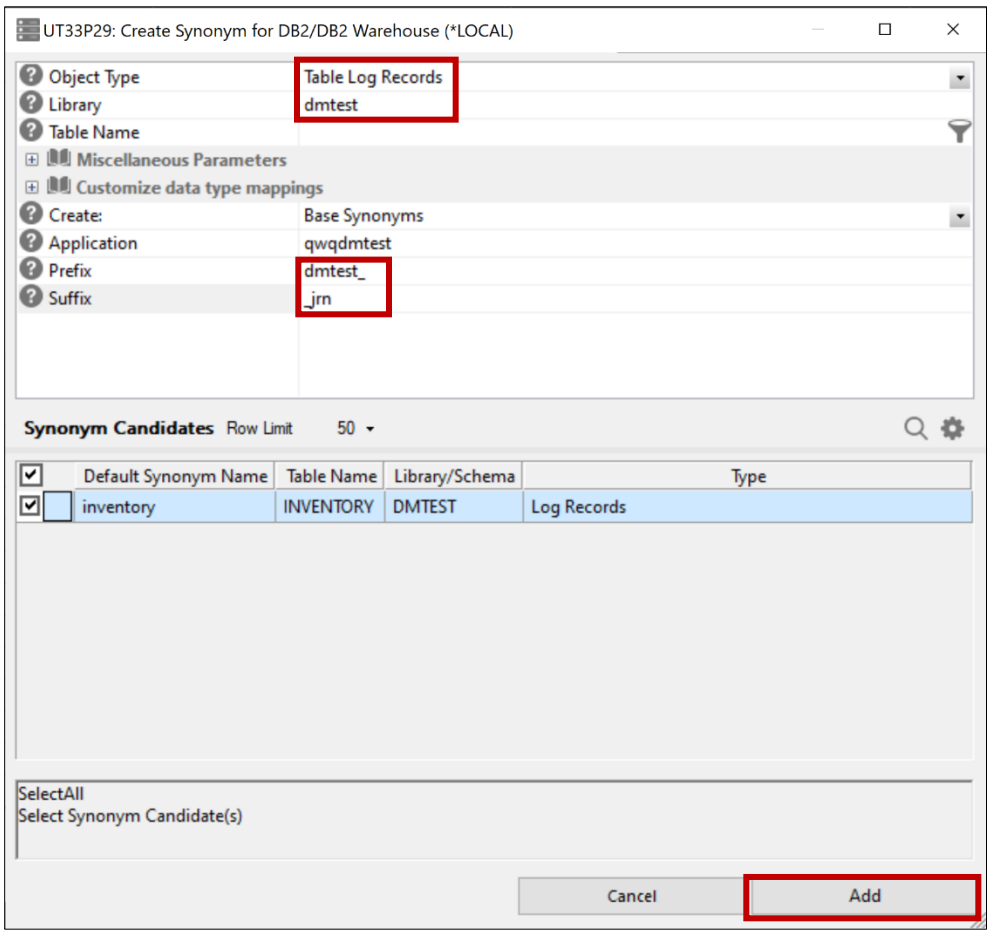


Figure 54 Creating a synonym over the remote journal

Double click on the newly created synonym `dmtest_inventory_jrn` to open it. Click the + next to `Variables` to expand the section. Right click on `&&CDC_LOG_NAME` and click `Properties`. For `Default Value`, enter `QSQJRN`.

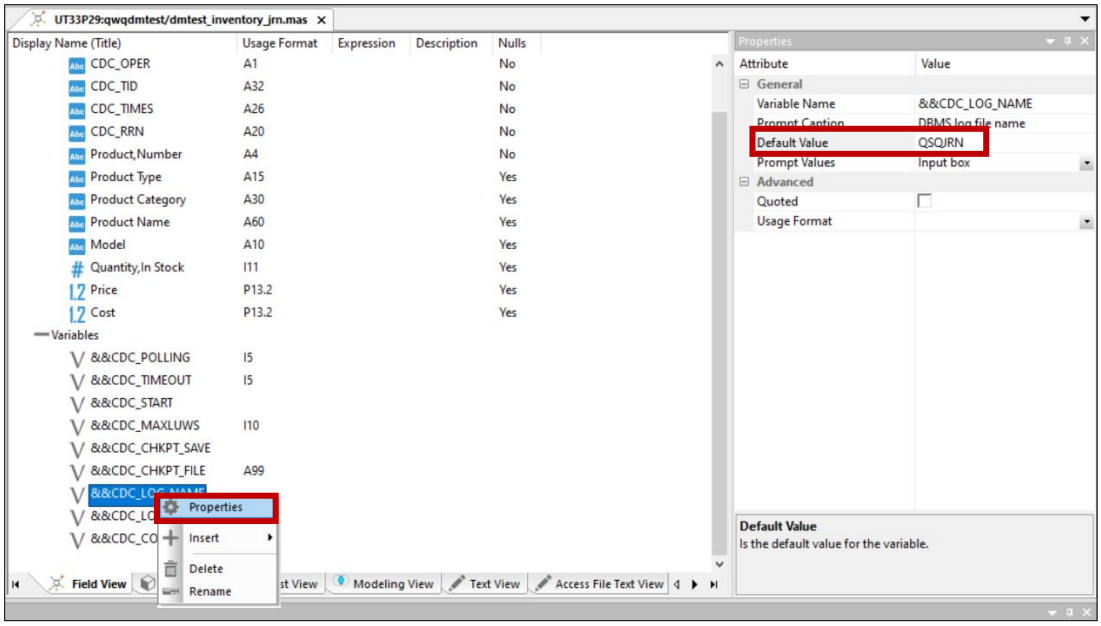


Figure 55 Editing the log name property

Now click on `&&CDC_LOG_LOCATION`. For `Default Value`, enter `DMTEST`. Then click `Save` to save the changes to the synonym.

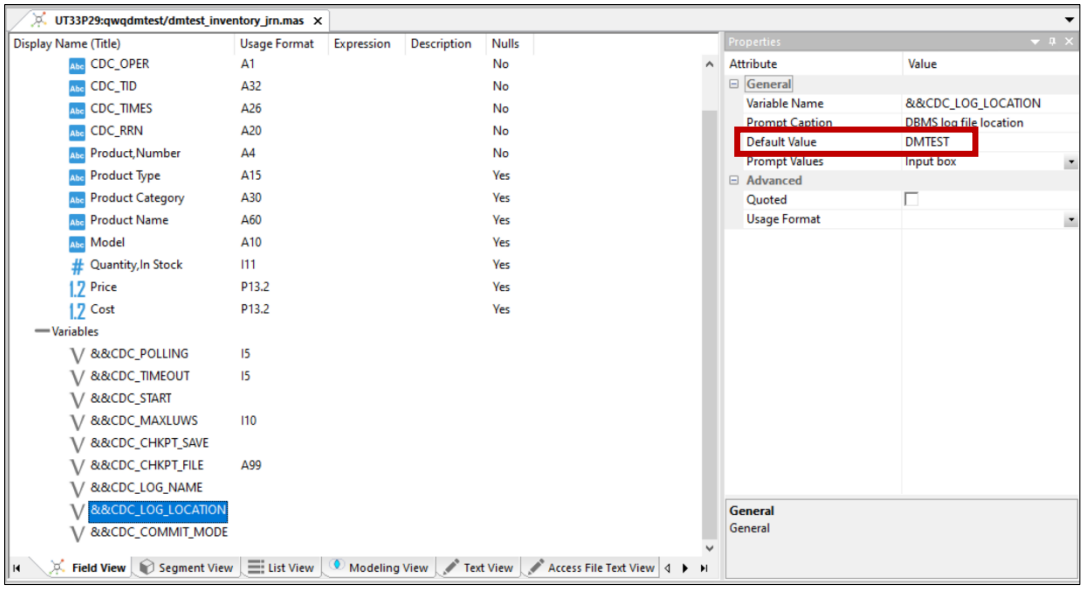


Figure 56 Editing the log location property

In the left navigation panel, right click on `dmtest_inventory_jrn` and click `Sample Data`. In the window, select all the fields and click `Sample Data`. You should see the rows of data originally added to the `DMTEST` table on the source system.

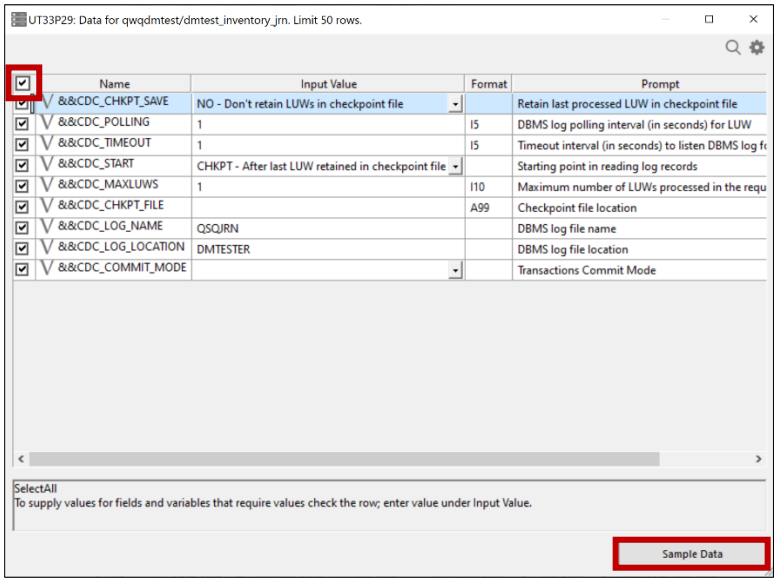


Figure 57 Sampling remote journal data

2.4.2 Creating a Bulk Load Flow

For this scenario, we will be creating and maintaining an inventory history table. This table keeps track of changing inventory daily and will allow us to track how inventory fluctuates over time. We will first create a 'bulk load' flow which will be run once to load a set of baseline data into the remote system. This baseline will be updated later when we receive changed data in the journal.

We need to create a synonym over the source data on the source system. Right click your test folder and click `New`, then `Synonym`. This time, the connection will be to your remote system. For `Object Type`, choose `Tables`. For `Library`, enter `DMTEST`. For `Prefix`, use `dmtest_` and for `Suffix`, use `_db2_sys` where `sys` is an identifier for your system. Then check the `inventory` table and click `Add`.

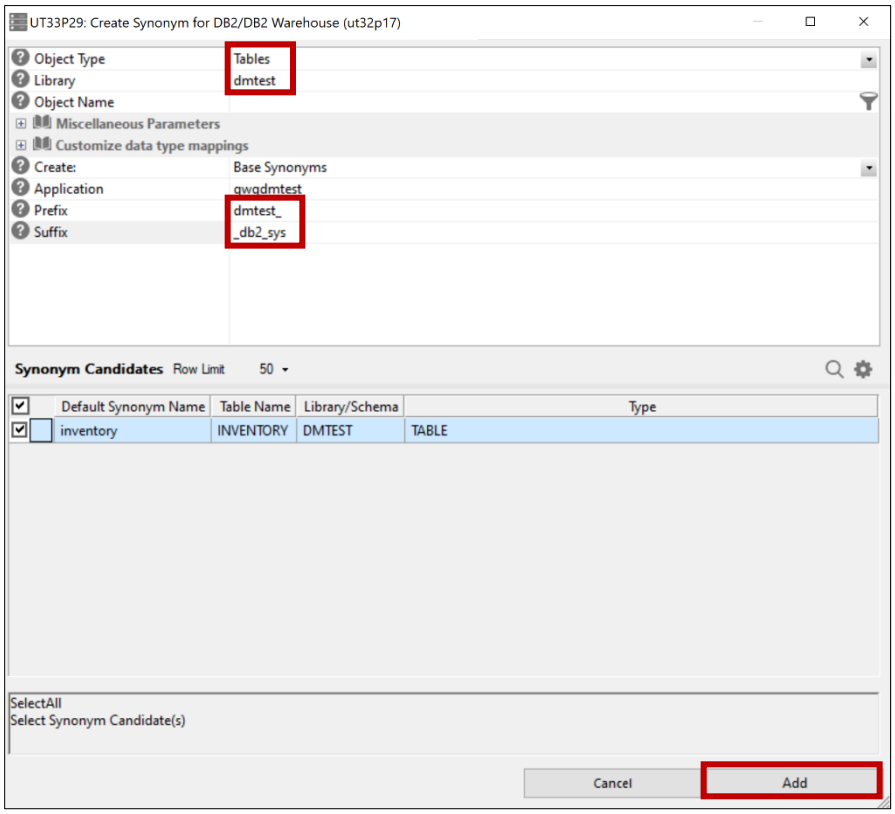


Figure 58 Creating a new synonym for the source data

Right click on your test folder and click `New`, then `Flow`.

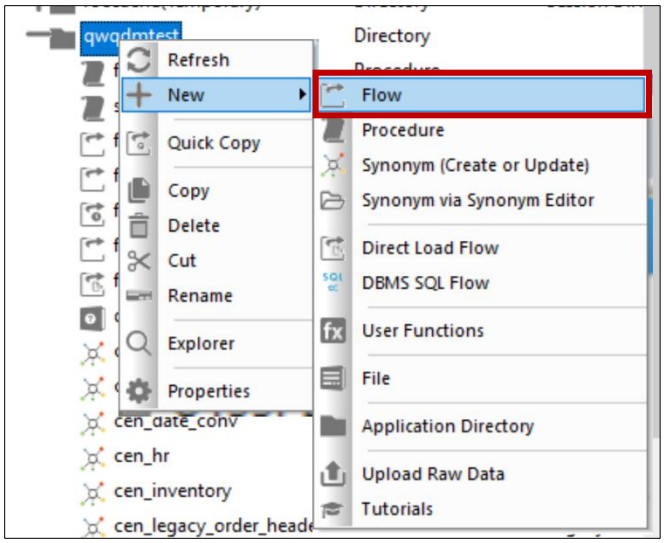


Figure 59 Creating a new flow

Drag the synonym `dmtest_inventory_db2_sys` into the palette to the left of the `SQL` icon. Then from the ribbon, drag a `New Target` into the palette to the right of the `SQL` icon.

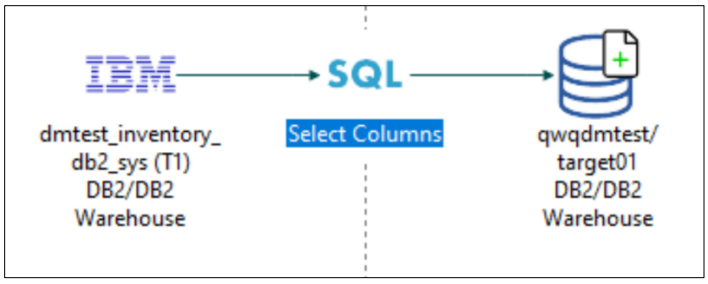


Figure 60 Inserting data sources and targets

Right click the `SQL` icon and click `Column Selection`. In the window, highlight all the columns on the left side and click the >> button to move them to the right.

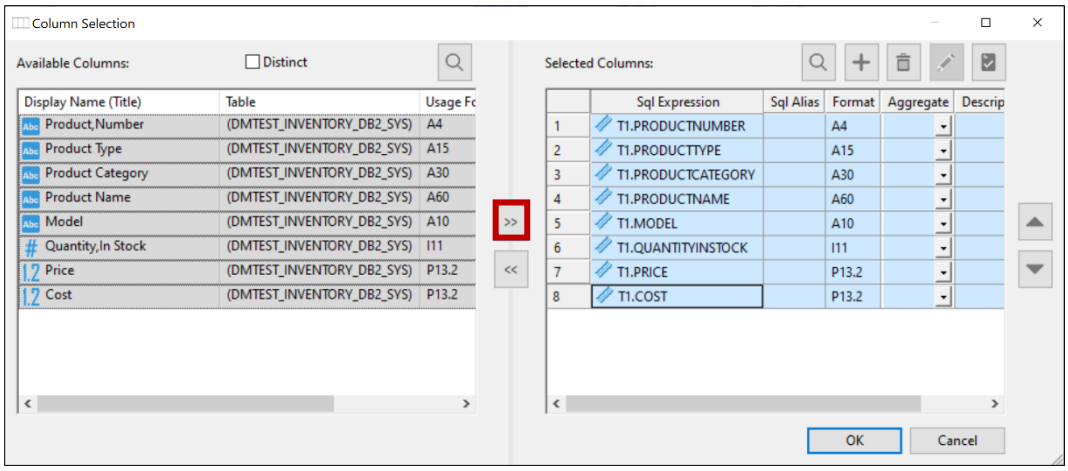


Figure 61 Selecting columns

Since we are creating an inventory history table, we need to add a date column. Click the `Insert Column` button to open the SQL calculator. Name the column `REFDATE`. For `Expression`, click the `Date` button and choose `Current Date`. Click `OK` to exit the calculator.

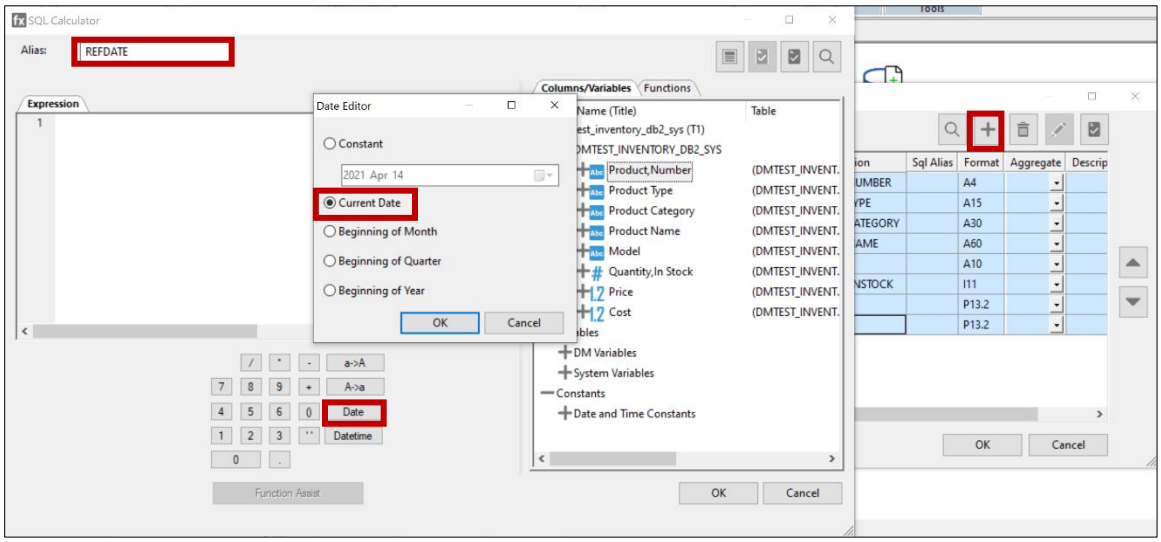


Figure 62 Creating a new column

Move the `REFDATE` column up to the top of the column list on the right side of the Column Selection window using the `up arrow` button. Click `OK` to exit the Column Selection window.

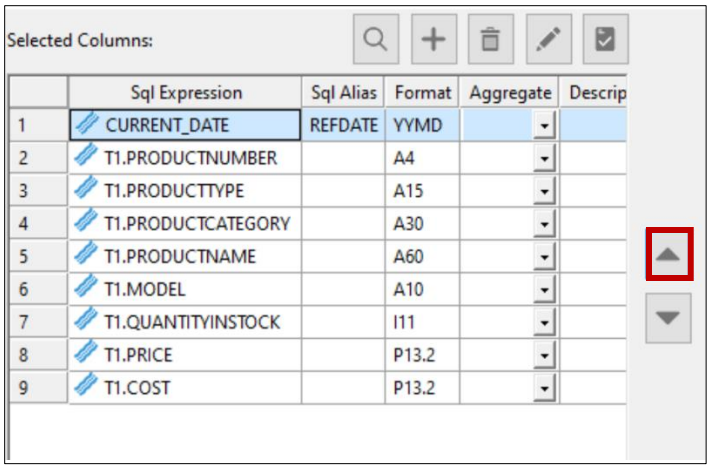


Figure 63 Moving REFDATE column to the top

Now right click on the target and click `Properties`. For `Synonym`, enter `qwqdmtest/inventory_history`. For Table, enter `dmtest/inventory_history`. For `Load Type`, choose `Insert Records From Memory`. Then click the ellipses (…) in the `Key` field and select `REFDATE` and `PRODUCTNUMBER` by clicking the >> button. Then click `OK`.

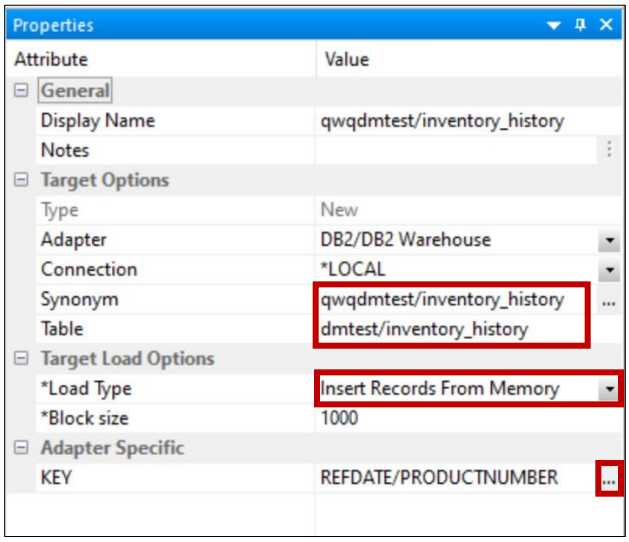


Figure 64 Setting the data target properties

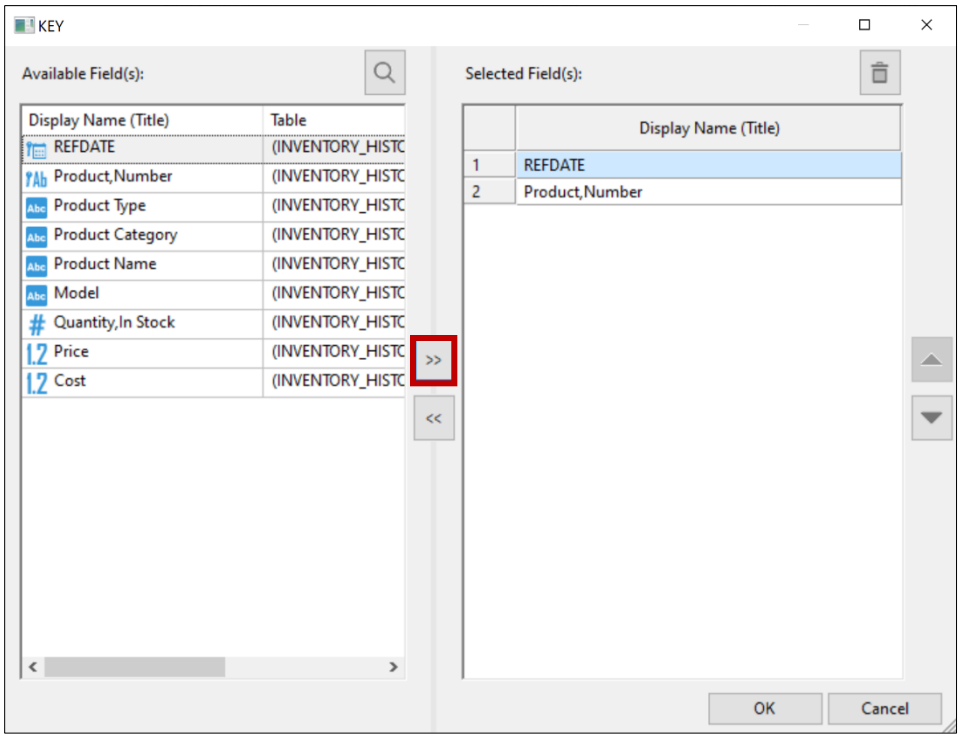


Figure 65 Selecting keys

Save the flow and call it `flow_bulk_load`. Now run the flow. Refresh your server in the left navigation tree and you will see that a new synonym called `inventory_history` has been created. This represents the target table which has been populated with all the records from the data source. In the console log, you can see the processing that took place when the flow was run.

2.4.3 Creating CDC Maintenance Flow

The second flow we will create will be a 'maintenance flow' that will be used to capture and apply changes that occur in the source tables after the 'bulk load' is run. Specifically, the flow will add the changes we receive from the remote journal to the local `inventory_history` table.

Right click on your test flow and click `New`, then `Direct Load Flow`. A direct load flow implies that data will come in and be processed immediately.

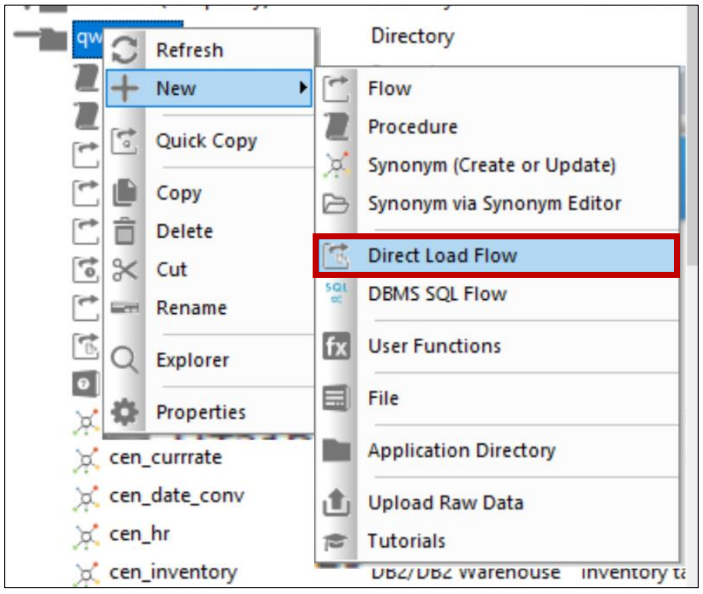


Figure 66 Creating a new direct load flow

Drag in the synonym for the remote journal into the left side of the palette. That is, `dmtest_inventory_jrn`. Then drag in the synonym for the existing target into the right side of the palette. That is, `inventory_history`.

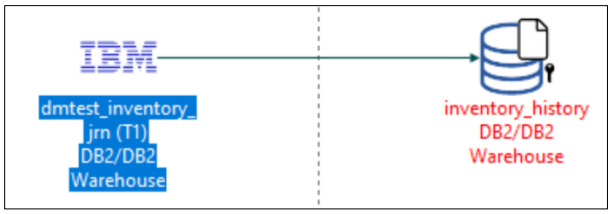


Figure 67 Adding the data sources and targets

Right click on the target and click `Target Transformations`. In the Transformations window, click the `Automap` button. This automatically maps columns in the source to columns in the target.

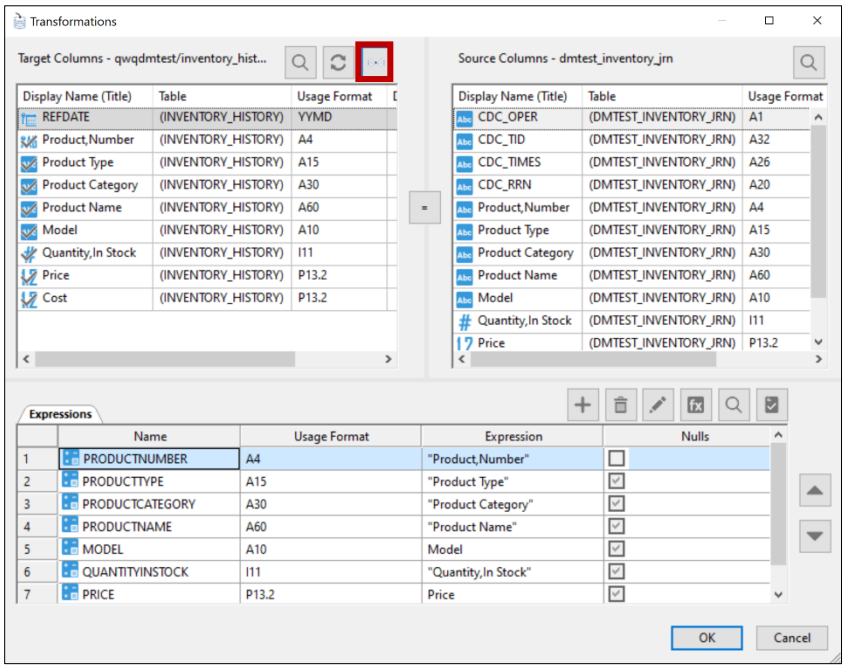


Figure 68 Automapping source columns

Now we want to add a current data field like we did for the bulk load flow. Click on the `Insert Immediate Transforms` button to open the SQL calculator. Name the new column `REFDATE`. In the `Expression`, click the `Date` button and choose `Current Date`. Click `OK` to exit the SQL calculator.

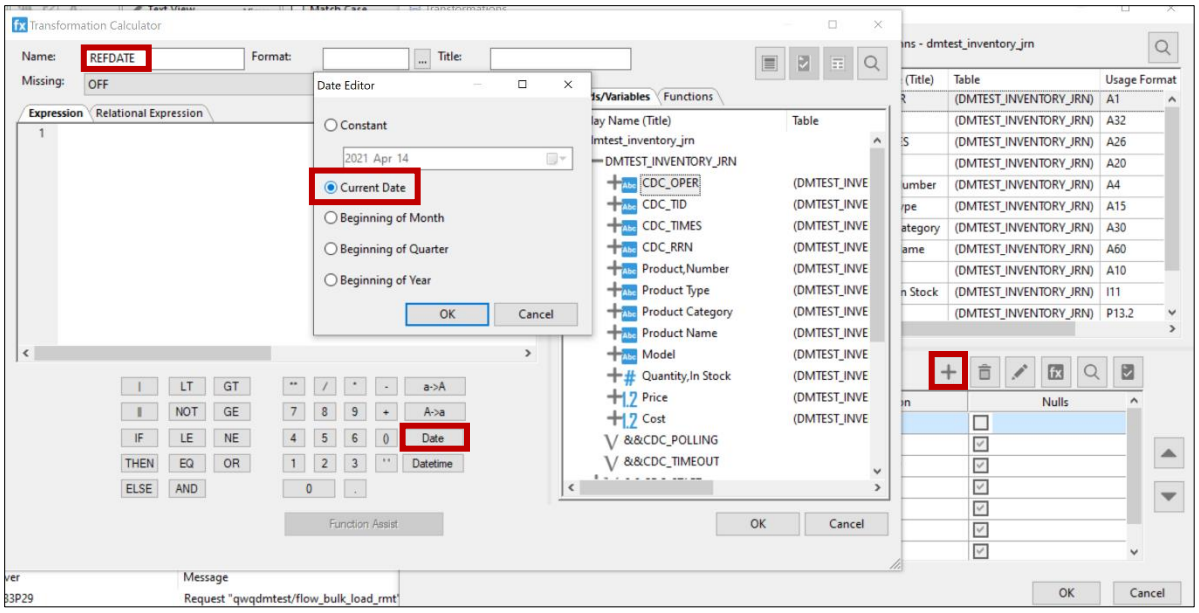


Figure 69 Inserting a new column in the target

Click the `up arrow` button to move the `REFDATE` column to the top of the Expressions list. Then click `OK` to exit the Transformations window.

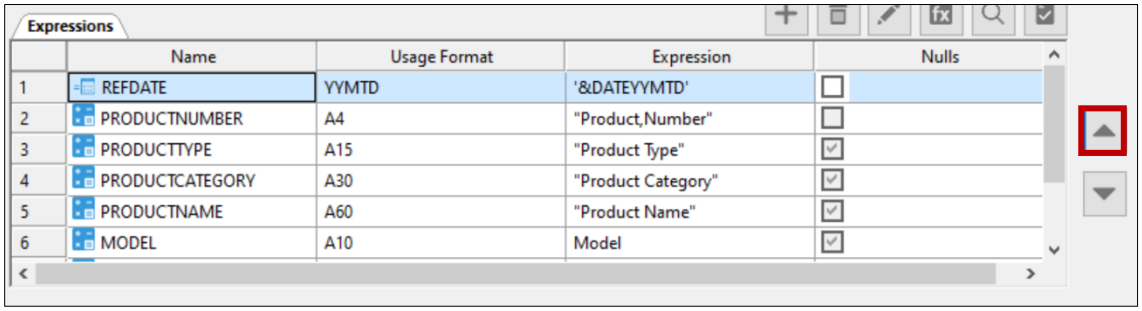


Figure 70 Moving the REFDATE column up

This flow will be capturing changes to the inventory table (via the journal) and putting it into the `inventory_history` table. We will be capturing the change once a day. However, for any given day it is possible that there could be multiple changes to a particular product in the inventory. As before, we will only keep one inventory change per day per product, we will keep the last change for a given day.

Right click on the target and click `Properties`. For `Load Type`, choose `Insert/Update`. For `If the record exists`, choose `Update the existing record`. For `If the record does not exist`, choose `Include the record`.

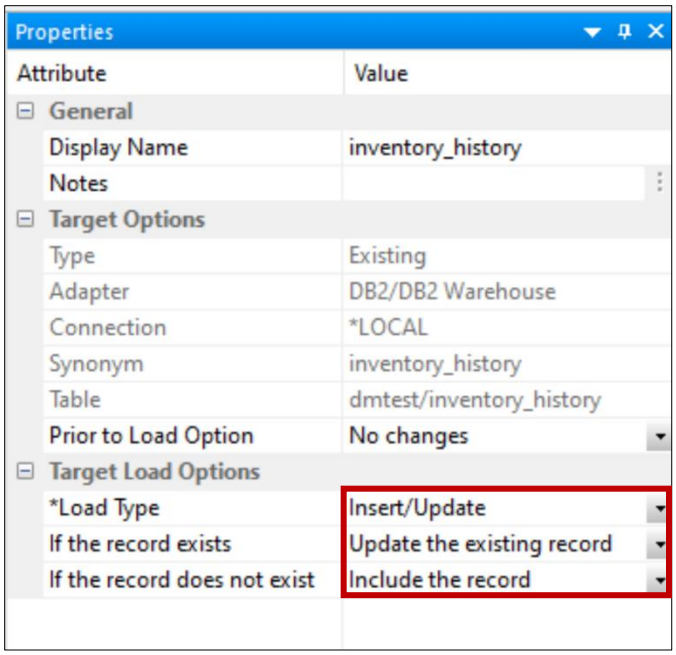


Figure 71 Changing the data target properties

Save the flow and name it `flow_delta_change`. Now run the flow. In the console log, you can see the processing that took place when the flow was run. Notice that after the first run, 76 rows were affected by the merge statement. This is because the journal contained all the changes since we created the `inventory` table. If you run the flow again, you will notice that no rows are processed. This is because the log checkpoint file is updated with information about the last journal entry read, so when the flow is run again it starts reading in the journal after the last position recorded in the checkpoint file. Since there were no changes since the journal was last read, no rows are processed by the maintenance flow. See section 7.3.4 for more information.

Appendix

DataMigrator contains more features than discussed in this guide. For more information, check out the help text and online User's Guide. They are available from the main DMC screen by selecting the arrow next to the information icon in the upper right.

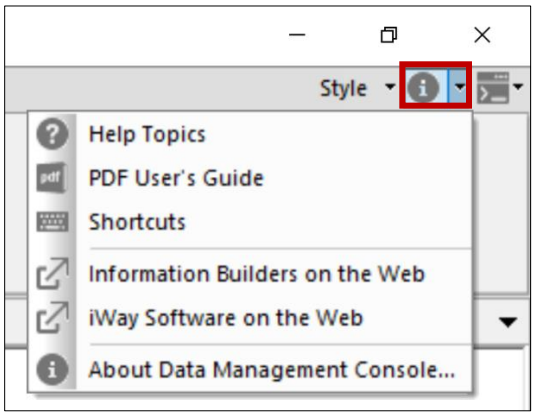


Figure 72 Online help information