DBMoto®

Setup Notes for IBM® Db2® z/OS Transactional Replications with Triggers

Version 9.5.0.20

Software Release Date: 4/23/2018

HiT Software, Inc.
4040 Moorpark Ave
Suite 221
San Jose, CA 95117

T +1 408-345-4001
F +1 408-345-4899
info@hitsw.com
www.hitsw.com

Document History

Version	Author	Date	Reviewer	Date	Approver	Date	Comments
1	JHLorenzin	6/13/18	ALaspertini		VFarruggio		

Table of Contents

Connection Type **1**

Trigger-Based Replication Overview **1**

DBMOTO Log Tables 2

Master table structure 2

Log Table Structure 3

Reading From/Managing Log Table 4

IBM Db2 LUW User Permissions **4**

Refresh with IBM DB2 z/OS as a Source Database 4

Transactional Replications/Initial Refresh with IBM DB2 z/OS as a Source Database 4

Enable Transactional Replication Wizard **6**

Log Type Screen 6

Trigger Settings Screen 6

The Ritmo/DB2 .NET Provider **7**

Enabling a trace 7

These notes provide essential information for setting up replications using **IBM Db2 z/OS** as a source database for one-way mirroring and synchronization**.** The setup process for a refresh replication can usually be completed using the DBMoto wizards without additional documentation because it does not involve access to the Db2 logs. For complete details on the setup process, check the *DBMoto User Guide* available from the Management Center **Help** menu or the *DBMoto Setup Guide*, available for download in the Help Center.

This guide describes the setup process using the Triggers option for one-way mirroring and synchronization when replicating data from a Db2 database. For mirroring and synchronization replications using Db2 z/OS as a source, DBMoto offers one approach:

- **Triggers**: Uses DBMoto triggers installed on the Db2 database to log changes

Connection Type

Ritmo/DB2 .NET Provider included with your installation of DBMoto.

Assembly: No value required

Trigger-Based Replication Overview

A database trigger is code that is automatically executed in response to certain events on a database table. To define a trigger-based replication (mirroring or synchronization), you need to provide information in the Source and/or Target Connection wizards so that triggers can be created to log table changes for replication.

For each table involved in the replication, DBMoto creates 3 triggers in the source table that fire when a specific event occurs on a record:

- INSERT trigger which fires when a new record is being inserted in the table
- UPDATE trigger which fires when a record is modified
- DELETE trigger which fires when a record is deleted

If the replication is later deleted, the triggers are removed by DBMoto. However, note that if you change a replication from mirroring to refresh, the triggers on the source table are not deleted. All transactions will continue to be recorded in the log tables. If you are not planning to reset the replication to mirroring, it is better to delete the replication, so that the triggers are removed, and create a new refresh replication.

Data retrieved using the triggers is stored in log tables that are specified in your Source/Target connection. The master log table can be an existing table or one created specifically to hold DBMoto information. It contains general information about the transactions, like user name, timestamp, table name. A log table (_DBM__LOG_x) is also created for each source table in the replication, and contains the data changes identified by the triggers, as well as trigger objects _DBM__TRG_OBJS.

Note that DBMoto does not create a tablespace. If you want to have a table space named DBMOTO, you must create it beforehand using a SQL tool. Run the appropriate SQL statement for your environment. For example:

CREATE TABLESPACE DBMOTO

When creating a connection, it is important to set the retention time to keep the log table size under control. The higher the value, the more data is kept in the log tables. Try to estimate the number of transactions occurring in all the source tables during a retention period and be sure that the database and table space have enough storage capacity for all those transactions. The DBReplicator (engine) cleans up the log tables periodically, based on the

retention setting in the connection dialog. If the engine is not running, the log tables are not cleaned up. This might create space problems in the database as the logs grow in size. If you stop the engine and you are not planning to run it again, be sure to remove all the mirroring synchronization replications.

In addition, if you have many table replications in a single group, using a single connection, all the replications share a master log table. Access to the log table for each source table can become a bottleneck if there are many transactions using the same master log and log tables. DBMoto may report errors about locked tables during replication. Although DBMoto is able to recover from these errors and continue replicating, a better approach is to prevent the errors by splitting the replications into multiple groups with multiple connections and multiple master log tables. First, create multiple source connections to the database. Use the Transaction Log Type field in the Connection Properties of each connection to open the Setup Info dialog and create a new master log table for each connection.

During replication:

- When a record is inserted in the source table, the INSERT trigger fires and inserts one record in the master table and one record in the log table associated with the source table. The record inserted in the log table contains all the original values of the INSERT statement.
- When a record is deleted from a table, the DELETE trigger fires and inserts one record in the master table and one record in the log table associated with the source table. The record inserted in the log table contains the key values of the deleted record.
- When a record is updated, the UPDATE trigger fires and inserts one record in the master table and two records in the log table associated with the source table. The two records inserted in the log table contain all the record values before and all the records after the update.

A transaction ID is saved both on the master records and log record to maintain a link between the transaction and the data changes for that transaction. See “DBMOTO Log Tables” for more details on the structure of the Master and Log tables.

Your system administrator needs to create and define appropriate table spaces and databases to hold the log tables. They should be large enough to handle the expected amount of replication data.

DBMOTO Log Tables

Log tables are used to record all data changes made to the source tables. They are populated by triggers that are fired when source tables are modified. Currently, log tables must reside in the replication source database. Note that log tables are created by DBMoto only if they do not already exist in the database.

There are two log tables associated with each replication: a master table, common to all replications using that connection, and a log table, one for each replication. The master table keeps track of all the transactions affecting the source tables and it records general transactional information.

Master table structure

- `TID DECIMAL(31,0) GENERATED BY DEFAULT AS IDENTITY:`
  Transaction ID number associated with each record data change (transaction)

- `SNAME VARCHAR(128):`
  Name of the schema the transaction was applied to.

- `TNAME VARCHAR(128):`
  Name of the table the transaction was applied to.

- `TTS TIMESTAMP:`
  Transaction timestamp indicating when the transaction was submitted to the system

- `TUSER VARCHAR(128):`
  Name of the user who executed the transaction

Log Table Structure

The Log table contains the actual data changes for a specific source table. Its structure depends on the structure of the source table.

- `__TID DECIMAL(31,0):`

  Transaction ID, link to the corresponding record in the master table

- `__OP CHAR:`

  Indicates the type of operation:

  'I' INSERT
  'D' DELETE
  'B' UPDATE: values before the update
  'A' UPDATE: values after the update

- `<Field list>:`

  All the columns of the source table with their original data type. For example if the source table was created as:

```
CREATE TABLE SOURCET
(ID INTEGER,
 FNAME VARCHAR(30),
 LNAME VARCHAR(50))
```

  The log table will have the following structure:

```
CREATE TABLE _DBM__SOURCET
(__TID DECIMAL(31,0),
 __OP  CHAR,
 ID    INTEGER,
 FNAME VARCHAR(30),
 LNAME VARCHAR(50))
```

Reading From/Managing Log Table

DBMOTO reads the log tables using the SELECT SQL statement. First, it queries the master table to see if new transactions came in since the last processed __TID. If transactions are found, DBMOTO queries the corresponding log table to collect the data changes and apply them to the target table.

The SELECTS on the master and log tables are sorted by the unique column __TID which ensures that all records will be read in the order that they were written. DBMOTO also uses the unique __TID column to keep track of the point where the last record was read and processed from the log tables.

DBMOTO provides options to manage log records that have been read and replicated. They can be deleted from the log table as soon as they are processed or a retention time can be set to leave this record in the log tables for the specified number of hours.

IBM Db2 z/OS User Permissions

When setting up replications that use IBM DB2 z/OS as a source database, you need to be sure that the user ID used for making connections to the database has sufficient privileges to complete all the operations required for DBMoto to perform a replication.

This section is organized by the type of replication you want to perform. It describes in detail all the user authorities that will be required during the setup and execution of replications.

Refresh with IBM DB2 z/OS as a Source Database

1. AUTHORITY TO CONNECT TO DATABASE
   This should already be granted when the user is created. However, here is the syntax, just in case:
   ```
   GRANT CONNECT ON DATABASE TO USER <UID>;
   ```

   Example where DBMOTO is the user ID:
   GRANT CONNECT ON DATABASE TO USER DBMOTO;

2. AUTHORITY TO SELECT TABLES
   DBMoto runs a SELECT statement to identify records to replicate. Therefore, the user ID used to make a connection must have adequate authority to run a SELECT statement for tables involved in replication. The command is not needed if the user is the owner of the table (i.e., created the table.)
   ```
   GRANT SELECT ON TABLE <TABLE> TO USER <UID>;
   ```

   Example where DBMOTO is the user ID and DB2ADMIN.EMPLOYEES is the table:

   ```
   GRANT SELECT ON TABLE DB2ADMIN.EMPLOYEES TO USER DBMOTO;
   ```

Transactional Replications/Initial Refresh with IBM DB2 z/OS as a Source Database

This section includes information on the minimum permissions needed to set up and run mirroring where IBM DB2 z/OS is the data source.

3. (OPTIONAL) CREATION OF TABLESPACE
   Note that DBMoto does not create a tablespace, but the user requires access to a tablespace. If you want to have a table space named DBMOTO, you must create it beforehand:
   ```
   CREATE TABLESPACE <TBLSPACE> IN <DATABASENAME>
   ```
   Example where DBMOTO_TBLSPACE is the name of the tablespace and TESTDB is the database.
   ```
   CREATE TABLESPACE DBMOTO_TBLSPACE IN TESTDB
   ```
4. CREATE USER FOR DBMOTO OPERATIONS AND TABLESPACE PERMISSIONS
   The DBA should create a user for DBMoto operations on the machine, then grant privileges to a specific tablespace with the command below. The DB2 user ID you are planning to use should have sufficient permissions to complete all operations in DBMoto: permissions to connect, select tables, insert/update/delete records and so on.
   ```
   GRANT USE OF TABLESPACE <TBLSPACE> TO <UID>;
   ```
   Example where DBMOTO_TBLSPACE is the name of the tablespace and DBMOTO is the name of the user.
   ```
   GRANT USE OF TABLESPACE DBMOTO_TBLSPACE TO DBMOTO;
   ```
5. AUTHORITY TO SELECT TABLES
   DBMoto runs a SELECT statement to identify records to replicate. Therefore, the user ID used to make a connection must have adequate authority to run a SELECT statement for tables involved in replication. The command is not needed if the user is the owner of the table (i.e., created the table.)
   ```
   GRANT SELECT ON TABLE <TABLE> TO USER <UID>;
   ```
   Example where DBMOTO is the user ID and DB2ADMIN.EMPLOYEES is the table:
   ```
   GRANT SELECT ON TABLE DB2ADMIN.EMPLOYEES TO USER DBMOTO;
   ```
6. AUTHORITY TO SET UP TRANSACTIONS
   To set up transactional replications, you need read-write access to the _DBM__TRG_OBJS table:
   ```
   GRANT TRIGGER, SELECT, INSERT, UPDATE, DELETE ON TABLE <DEFAULT_SCHEMA>.
   "_DBM__TRG_OBJS" TO <UID>;
   ```
   Example:
   ```
   GRANT TRIGGER, SELECT, INSERT, UPDATE, DELETE ON TABLE DB2ADMIN.
   "_DBM__TRG_OBJS" TO DBMOTO;
   ```
7. AUTHORITY TO ACCESS TRANSACTION LOG
   You need to access the transaction log information stored in the table _DBM__TRG_OBJS. Note that if you revoke read access, the replication works but it is unable to read the last ID.
   ```
   GRANT SELECT ON TABLE <DEFAULT_SCHEMA>."_DBM__TRG_OBJS" TO <UID>;
   ```

Example where the user ID is DBMOTO:

```
GRANT SELECT ON TABLE DB2ADMIN."_DBM__TRG_OBJS" TO DBMOTO;
```

Enable Transactional Replication Wizard

For transactional replications (mirroring and synchronization), use the Enable Transactional Replication wizard after setting up a source connection. The following field(s) require specific information for IBM Db2.

Log Type Screen

The only option available for IBM DB2 for z/OS is Triggers.

Trigger Settings Screen

Master Table

Either specify an existing qualified table name, or click Change to create a new table to hold general information about replication transactions including user name, timestamp, table name for each transaction.

There are two tables associated with each replication: a Master table, common to all replications using that connection, and a Log table for each replication source table. The Master table keeps track of all the transactions affecting the source tables and it records general transactional information.

Master and Log tables are created in the schema specified when you set the Master table name. You can choose a Master table name, or use the default _DBM__MASTERLOG. Log tables are automatically generated by DBMoto and the names are _DBM__LOG_#, where # is a number. The selected schema for the Master and Log tables must not contain other non-DBMoto tables with names _DBM__LOG_# . HiT Software recommends that you create a new schema to use specifically for the DBMoto Master and Log tables.

Tablespace

HiT Software recommends that you assign a tablespace for the Master table and Log Tables so that it is easier to control log table sizes. If you leave this field blank, the default tablespace value for your login ID will be used. Your system administrator should be able to provide you with the appropriate value for this field.

If you are using IBM Db2, the value can be entered as dbname.tablespace.

Retention Time

The amount of time in hours that a transaction is kept in the log tables. The default value is 72 hours. When the amount of time a transaction resides in the log exceeds the retention time, the transaction is permanently removed from the log tables. Tuning the retention time provides control over the size of the log tables. It is also possible to instruct DBMoto to remove all the processed transactions at the end of each mirroring interval. Tuning the retention time provides control over the size of the log tables.

Delete Block Size

Based on the retention time, DBMoto deletes items from the log. This field specifies the maximum number of records to delete from the DBMoto log tables with a single SQL statement. The default value is 10,000 records. You do not typically need to edit this value.

Lower-case Trigger Identifiers

Check this option if your database installation uses lower-case trigger identifiers.

Trigger Order

Always inactive for IBM DB2 for z/OS sources.

The Ritmo/DB2 .NET Provider

The Ritmo/DB2 .NET provider is included with your DBMoto release and installed as part of the DBMoto setup. The version of Ritmo/i that is provided with DBMoto does not include the Ritmo Toolbox or the developer tools. For more information about the different Ritmo versions available, check the BackOffice Associates web site.

The Ritmo/DB2 files can be found in the DBMoto installation folder, in a separate folder called **Ritmo_DB2**.

The following topic may be useful when using Ritmo/DB2 with DBMoto.

Enabling a trace

The file Ritmo_i.xml contains configuration settings for Ritmo and can be found in the folder where Ritmo was installed. You can set a trace file name and enable the trace from this file. Note that the trace will run whenever Ritmo is in use, and you should set <traceflag> to False immediately after completing the operations that you wanted to trace. If you leave the trace running, it can affect performance and build up large trace files.

1. In the Windows Explorer, go to the DBMoto installation folder, then to the Ritmo_DB2 folder.
2. Open the file Ritmo_DB2.xml in a text editor.
3. Modify the trace entry (in bold below) as needed:

```
<?xml version="1.0" encoding="utf-8" standalone="yes" ?>
<configuration>
<configSections>
<section name="trace" />
</configSections>
<trace>
<tracefile>C:\Program Files\HiT Software\DBMoto V6\Ritmo_i\logs\Ritmo_Db2.trc</tracefile>
<traceflag>True</traceflag>
</trace>
</configuration>
```

4. Save the file and exit the editor.

Last Updated on 9/13/2018