SQL-Sync

Presented by: Luis Gomez

The Mastermind Series

Presented by VERTICAN TECHNOLOGIES

SQL-SYNC

- What is it?
  - A set of tools that copy most of the data from Collection-Master to a SQL Server database.
  - The SQL Server data will be updated “near real time” with new records added and existing record refreshed periodically.
- Why do I want this?
  - Having Collection-Master in SQL allows users to query and leverage data.
- How does it work?
  - By running a series of SYNC sessions that run based on a schedule. Each session is assigned a list of jobs to Bulk Insert or Incremental Sync the database.

SQL-SYNC

- SQL-Sync is an extremely powerful add-on module to Collection-Master, designed to manage and automate the process of synchronizing the Collection-Master database with the SQL Server.
- Click to Buy – It’s Free!
- Once purchased, the Implementation Team will reach out and help you set up the product.

SQL Server

- Microsoft SQL Server is required for SQL-Sync
  - Microsoft recommends that the SQL Server be a dedicated machine or VM on the system.
  - Use the SQL server for many databases:
    - vMedia
    - Collection-Master SQL-SYNC
    - Q-LawE
    - Others
- SQL Server DBA (Database Administrator)
  - SQL server requires a DBA to maintain, configure, tune performance, backups, etc.
  - Many firms will outsource the DBA role and only require part-time services.
- Vertican Implementation
  - 4 CPU's
  - 14 GB of Memory
  - This is a *tiny* SQL Server
    - Add CPU & memory to improve performance.

Collection-Master Server

- Must be run in Client-Server mode
- Secure Data is supported and encouraged
- Vertican Implementation
  - 8 CPU's
  - 32 GB of Memory
  - This is a *tiny* CM Server
    - Add CPU & memory to improve performance
    - Plan on adding 1 CPU/core per SQL-Sync Session (Minimum of two sessions)
    - Generally speaking, 64GB is the minimum memory recommended.

SQL-SYNC Implementation

The Implementation Team Process:

- 1st call
  - Discuss what is needed prior to set up and schedule.
- 2nd call
  - Install/Setup
  - Initiate bulk sync
- 3rd call
  - Walkthrough of how to set up sync schedule
  - Add additional sync session batch files
  - Answer any final questions
- E-Mail throughout. Once implementation process has begun, you can e-mail the team between sessions if any questions arise.

SQL-SYNC Implementation

DIY – You can **do it yourself!**

- Within CM
  - Help → Help Manuals → SQL-Sync manual
    - SQL-Sync Set up: Step-by-Step Guide
- Read the SQL-SYNC manual (all of it)
  - The manual covers details about the product and process.

SQL Server Database

- On the SQL-Server
  - Create a database with a unique name that corresponds to the Collection-Master system you are trying to sync.
  - Set users up for this database, two basic users are required.
  - Admin user: To perform all the administrative functions on this database (SQL db_owner).
  - Functional user: To perform all the day-to-day DB functions such as creating /deleting/writing to tables (SQL ddl_owner)
    - Note: SQL-Sync only serves to maintain the SQL Database, functional users are users that will consume the SQL Server Database.

Mapdrive._CS

- SQL-Sync-SERVER = Server Name (Ex: CM-SQL)
- SQL-Sync-DATABASE = Database Name (Ex: CM)

```
Rem [---- Start of SQL-SYNC-SETUP ----]CRLF
SETENV SQL-SYNC-SERVER [SQL-SERVER]CRLF
SETENV SQL-SYNC-DATABASE [SQL-DATABASE]CRLF
Rem [---- End of SQL-SYNC-SETUP ----]CRLF
```

SQL Sync Tables

- UPPER CASE tables
  - Collection-Master data or tables.
- Mixed Case Tables
  - Configuration or calculated data provided by SQL-Sync
  - CategLst
  - Folders
  - SyncItemizedLog
  - SyncLog
  - SyncSchedule
  - SyncSQLLog

SQL Sync Tables

- CategLst
  - Itemized List of Financial Codes
  - COSTCODE – Table that contains the various Cost Codes set up in [2-S-2].
  - LETTERS – Table that contains the various Documents set up in [1-7-1].
- Folders
  - 1 = Open Claims Information
  - 2 = Closed Claims Information
  - 10 = Common Data Files
  - 11 = Share Data File
  - 20 = EDI Data Files
  - 30 = HELP Files
  - 40 = ZipCodes
  - 1000+ = Bank Accounts

SQL Sync Tables

Field Name	Description
[LogId]	Identity Field (Counter)
[UserName]	Windows Login Name
[ClientWorkStation]	Workstation running Sync
[BRSession_Id]	Session ID
[EventId]	Event ID from SyncSchedule
[TableName]	SQL name for table being updated
[Folder_Code]	DATA, HISTORY, etc.
[CallingFunc]	BULK_INS, INCR_SYNC, SYNC_FOLDER
[StartTime]	Time process started
[EndTime]	Time process ended
[BRTotal_Recs]	Total # of Records (File size)
[AddCount]	# of Records Inserted
[DeleteCount]	# of Records Deleted
[UpdateCount]	# of Records Updated
[SqlSession_Id]	SQL Session ID (1,2,etc)

- SyncItemizedLog
  - Contains itemized details on each SQL Job.

SQL Sync Tables

- SyncLog
  - Used to record the last time each event was run

Field Name	Description
[EventId]	Event ID from SyncSchedule
[LastRunAt]	The last time the Event was run

- SyncSchedule

Field Name	Description
[EventId]	Identifier for each Event (Unique ID)
[Event]	Actual Event including table (BULK_INS:, INCR_SYN:)
[Folder_Code]	Code to identify folder (Open,Closed, Account, Misc, EDI, ZipCodes, SQL)
[Group_Code]	English description for the Table/Group of tables
[TimeOfDay]	Mostly unused enter a military time to execute once per day
[FreqInMinutes]	How long to wait in minutes between running this event. (0=Instant)
[Start_Record]	Very large files can take too long to run INCR_SYN. Starting Record
[End_Record]	Ending Record (- values are relative to last record)
[Start_TimeOfDay]	Minimum time for session to run. Low priority files can be run in off hours.
[End_TimeOfDay]	Maximum time for session to Run. Low priority files can be run in off hours.
[Enabled]	True = Run | False = Don't Run
[Session_ID]	Session # - When running session in CM, each session may pick a #.

How to Set Up Sync-Schedule.

- Dbo.Synschedule
  - This table is delivered with a possible configuration
  - Session 1: Bulk_INS Imports new Records.
    - Bulk_INS is very fast
    - Tables scheduled every 60 minutes from 06:00 to 20:00
    - Except closed BUCKETS_EDI & FINAN_EDI
  - Session 2: Inc_Sync Updates existing records.
    - Inc_Sync takes a lot more time.
    - Highest priority scheduled every 60 minutes from 06:00 to 20:00
    - Other items scheduled every 60 minutes from 18:00 to 20:00
  - Tables that need to be updated more often, create additional sessions.
    - It’s a balance of time & resources. Running nine sessions updating every five minutes requires very beefy CM & SQL servers!

SQL Sync Tables

- SyncSQLLog
  - Itemized Log of Schema updates

Field Name	Description
[TransId]	Identifier for each Log Entry (Unique ID)
[StartedAt]	Start Time for Log Entry
[EndedAt]	End Time for Log Entry
[Module]	FN_ALTER_TABLE, FNTRUNCATE_TABLE,FN_CREATE_SQL_TABLE)
[SqlString]	Actual SQL applied to database

SQL Sync Views

- View_Paperless
  - View that performs the necessary joins and concatenation to display the paperless file as it displays in Collection-Master.

Field Name	Description
[Folder_Number]	(1) Open / (2) Closed
[Record_Number]	Record # in CM (Natural Order)
[FILENO]	Claim # in Collection-Master
[DATE_TIME]	Date & Time of Transactions
[INIT]	Use Initials
[CODE]	Paperless File Code
[NOTE]	Paperless File Note (Including Translated Description)
[BILLED_DATE]	Date associated with Document Billing
[RECNO]	The Record # used for linking with tables like PS Comments
[RECEIVED]	Financial - Amount Received
[DISBURSED]	Financial - Amount Disbursed

SQL Sync Views (Other)

- View_Schedule
  - Links View_Sync_Schedule & View_SyncLog
  - For each Event, shows the Last Time it ran, and the next scheduled run
- View_SyncItemizedLog – Itemized list of sync sessions
- View_SyncLog
  - Links SyncLog & SyncSchedule
  - Depreciated, use View_Schedule instead
- View_SyncSchedule – Itemized list of sync schedule

Scalar-Valued Functions

- **ufnCateg2**
  - Converts a Paperless Code to the Description
  - `Select dbo.uFNCateg2('51') as Description`
    - Returns ‘File Suit’
- **ufnCollord**
  - Expands a Collection Order uses default if blank
  - `Select dbo.ufnCollord('CPI','COPLVIE’)`
    - Returns ‘CLIPVI’
- **ufnDate_Time**
  - Merges a Date field with a numeric “100’s of Seconds” as stored in DB
  - `Select [dbo].[ufnDate_Time]('2021-03-03',105100)`
    - Returns ‘2021-03-03 00:17:31.000’
- **ufnSeconds_Spent**
  - Compares two time fields and returns “100’s of Seconds” as stored in DB
  - `Select [dbo].ufnSeconds_Spent('08:00:00','11:30:00’)`
  - Returns 12600
- **ufnStime$**
  - Converts ‘100’s of Seconds’ to “Normal Time”
  - `Select [dbo].ufnStime$(12610)`
  - `Select [dbo].ufnStime$([dbo].ufnSeconds_Spent('08:00:00','11:30:00'))`
  - Returns’ 00:02:06:1’ HH:MM:SS:MS

SQL Tips

- WITH (Nolock)
  - Prevents query from locking tables.
  - `FROM [CM-SQL].[dbo].[TRACKUSR] WITH (NOLOCK)`
- COUNT(*), SUM, MAX, MIN /w GROUP BY
  - Count , Add, largest, smallest
- GETDATE()
  - Returns Today. >=GETDATE()-30 is 30 days or newer.
- CAST
  - Converts data types
  - `CAST(dbo.ufnSeconds_Spent([sTime],[Time]) as INT)`
- ISNULL
  - Used to handle NULL values.
- https://www.w3schools.com/SQL/
  - Everything you always wanted to know about SQL!

dbo. AccountCard

- Stp_Make_AccountCard
  - Step 1 Creates dbo. AccountCard
  - Run this one time
- Stp_Update_AccountCard
  - Step 2, populates dbo. AccountCard with account card information
  - Query may be a bit slow as it calculates the account card and updates every claim.
  - The Resulting table includes many Collection-Master calculated field.
  - Look at the provided stored procedure to learn how Collection-Master internals work!

Connecting to SQL Server in MS-Excel

```
SELECT COUNT(*) as Count,
  SUM(CAST(dbo.ufnSeconds_Spent([sTime],[Time]) as INT)) as Spent
      ,[WHOAMI]
      ,[TRCK_DATE]
FROM [CM-SQL].[dbo].[TRACKUSR] WITH (NOLOCK)
  Where TRCK_DATE>=GETDATE()-30
  GROUP BY WHOAMI,TRCK_DATE
  Order by TRCK_DATE,WHOAMI
```

Importing Data into Collection-Master

- List of Claims
  - Query Claims
  - Report Generator
- Merge-POP
- CM EDI
  - Create
  - FILENO
  - PRE_J_RATE
  - POST_J_RATE
  - INT_DATE
  - STORED_INT
  - #

CM EDI RT 171

```
-- Stop Accruing Interest Using the Last PER_DIEM_INT Value.
-- Select Claims in Lawson, AR

SELECT
'171' AS Record
,'D' as [H]
,[MASTER].[FILENO]
,'0.00' as PRE_J_RATE
,'0.00' as POST_J_RATE
,CAST(GETDATE() as DATE) as INT_DATE
,PER_DIEM_INT as STORED_INT
,D1_CS
,'#' as [#]
```

```
FROM [CM-SQL].[dbo].[MASTER] WITH (Nolock)
  INNER JOIN [CM-SQL].[dbo].[DEBTOR] With (Nolock)
  ON MASTER.FILENO=DEBTOR.FILENO
    and DEBTOR.NUMBER=1
Where Master.Folder_Number=1
    and DEBTOR.ST = 'AR'
    and DEBTOR.CITY= 'Lawson'
```

Link SQL to MS-EXCEL

- Create a View in SQL-Server
- Link Excel to SQL Server Database
- Export as Text File
- Text File may be used in Collection-Master
- CM EDI 171 sample
- Merge-Pop, ASCII list of Claims, GENERAL EDI

Link SQL to MS-ACCESS

- Create a View in SQL-Server
- Link Excel to SQL Server Database
- Export as Text File
  - MS-Access has a Bug, [#] Exports as a . *(have to fix manually)*
- Text File may be used in Collection-Master
- CM EDI 171 sample
- Merge-Pop, ASCII list of Claims, GENERAL EDI


Q
A

The Mastermind Series

To learn about upcoming trainings:
https://vertican.tech/mastermind/

To view past trainings:
https://vimeo.com/ondemand/verticanmastermindseries/