

Cloudbridge DM Batching and Non-Batching

Technical Guide

@Copyright 2022 Version 1 – All Rights Reserved

# 1. Overview

This section provides a high-level summary of the recent ETL process enhancement, highlighting both Batching and Non-Batching approaches.

## 1.1. Batching

## Batching, within the context of ETL (Extract, Transform, Load), involves collecting and processing data in chunks or batches rather than individually. This optimization enhances processing efficiency and resource utilization. The efficiency gains arise from reduced overhead associated with transactional operations when dealing with records collectively. Additionally, batching is crucial for managing substantial data volumes, allowing for parallel processing and horizontal scalability to accommodate increased data loads.

## 1.2. Non-Batching

Non-Batching describes a data processing method wherein individual records are processed one at a time without being grouped into batches. This implies that each record undergoes independent extraction, transformation, and loading, rather than being part of a larger collection. For scenarios with relatively small data volumes and manageable overhead in processing individual records, a non-batching approach may suffice.

# Design Flow

This section outlines the Design Flow Diagram of Maximise DM Batching and Non-Batching process, illustrating how the data is displayed according to the provided input while Extracting into stg tables.

A diagram of a flowchart

Description automatically generated

# Configuration

The following configurations must be set up for both Batching and Non-Batching.

## 3.1. Batching

* Verify that Batch\_Load column is ‘Y’ in the **XXMX\_MIGRATION\_METADATA**  table.
* For Finance, the details regarding batching is stored in the **LOAD\_BATCH** column of both the STG and XFM tables.
* Whereas for HCM, the details regarding batching is stored in the **BATCH\_NAME** column of both the STG and XFM tables.
* Ensure that the **COMMON\_LOAD\_COLUMN** in the **XXMX\_XFM\_TABLES** is updated for the corresponding business entity and sub-entity.
* Utilize the **Lookup\_Data\_Migration\_Extracts** OIC lookup to specify the desired batch count of records in the extract\_batch\_count field for extracting data in batches.
* Ensure that the below tables are populated.
* XXMX\_STG\_TABLES
* XXMX\_STG\_TABLE\_COLUMNS
* XXMX\_XFM\_TABLES
* XXMX\_XFM\_TABLE\_COLUMNS

## 3.2. Non-Batching

* Verify that Batch\_Load column is ‘N’ in the **XXMX\_MIGRATION\_METADATA** table.
* Ensure that the below tables are populated.
* XXMX\_STG\_TABLES
* XXMX\_STG\_TABLE\_COLUMNS
* XXMX\_XFM\_TABLES
* XXMX\_XFM\_TABLE\_COLUMNS

# Technical Specification

This section outlines the technical specifications for batching and non-batching.

# 4.1. Batching

## 4.1.1. Extract

During the extraction phase, batching for the extraction process is enabled and initiated if **BATCH\_FLAG** is set to 'Y' based on the data from **xxmx\_migration\_metadata** for the corresponding Business Entity and Sub entity.

* To determine the desired batch count of records for extracting data in batches, the batch count is retrieved from the OIC lookup named **Lookup\_Data\_Migration\_Extracts**, specifically from the extract\_batch\_count field.
* The extraction process triggers **xxmx\_utilities\_pkg.batch\_extract\_data**, considering parameters such as the application suite, business entity, sub-entity, batch count, and phase. After the extraction is completed and data being transferred to STG tables.
* During Extraction for first sub entity sequence from xxmx\_migration\_metadata table, if the flag is set to ‘Y’, it calls **batch\_extract\_data** procedure from **xxmx\_utilities\_pkg** to batch the first sub entity sequence.
* Once the formation of batches is completed for the first sub\_entity\_seq, the 'Load\_batch' column for finance and 'Batch\_name' column for HCM will be updated with the batches in the STG tables. Details of batches is stored in **XXMX\_DM\_FILE\_BATCH** table.
* For other sub entity sequences during their extraction, it checks if sub\_entity\_seq1 is batched, only then batches the remaining sub\_entity sequences according to the common\_load\_column set in xxmx\_xfm\_tables.
* If sub\_entity\_seq1 is not batched, an exception is thrown in the xxmx\_module\_message table, indicating that batching must be completed for sub\_entity\_seq1 before proceeding further. Until this is done, further batching cannot be processed.
* The table below provides a list of tables, procedures, and description for each detailing their functionalities.

|  |  |
| --- | --- |
| Key | Description |
| BATCH\_EXTRACT\_DATA | This procedure handles the batching of extracted data. |
| XXMX\_DM\_FILE\_BATCH | This table stores information such as the table name, batch name, batch column, and sequence batch of the sub\_entity\_sequence. |
| XXMX\_MIGRATION\_METADATA | Table holds details of Business Entity and Sub Entity. Extract packages, Staging, Transformation and External Tables. Client can enable and disable any business entity or sub entity for Extract, Transform or Load. |
| XXMX\_MODULE\_MESSAGES | This table holds log messages for each phase – Extract, Transform and Load |
| XXMX\_UTILITIES\_PKG | Maximise Core utility package |
| XXMX\_DYNAMIC\_SQL\_PKG | Maximise Package for Transformation, Extract and Load |
| XXMX\_FIN\_STG\_EXTRACT\_PKG | Maximise Generic Package for Finance Extract and Transformation. |
| XXMX\_HCM\_STG\_EXTRACT\_PKG | Maximise Generic Package for HCM Extract and Transformation. |
| XXMX\_FUSION\_LOAD\_GEN\_PKG | Maximise Generic Package for Load file generation. |
| LOAD\_BATCH | This column holds the batched records for Finance in stg and xfm tables. |
| BATCH\_NAME | This column holds the batched records for HCM in stg and xfm tables. |
| BATCH\_LOAD | This column holds the flag for batching and non-batching in xxmx\_migration\_metadata table. |

## 4.1.2. Transform

The Transform approach remains consistent, with data being moved from STG to XFM through transform routines in alignment with the batching processes carried out during the transform phase.

## 4.1.3. Load

Following the completion of the extract and transform stages, the subsequent phase entails batching for the load process.

1. **Load Process:**

* The process starts by loading data into temporary tables ('xxmx\_csv\_file\_temp' for finance and 'xxmx\_hdl\_file\_temp' for HCM), and then recording this information in the 'XXMX\_LOADFILE\_STATUS\_LOG' table with a Load file ID and status set to 'R'. Subsequently, the data prepared in these temporary tables is ready for file generation on the server.

2. **Load File Generation in DB Server:**

* Load file generation on the DB server is managed by a scheduled integration in OIC named **'INTDM999015 DM Write Load File to DB Server (1.0)'**, running every 10 minutes. This integration selects Load file IDs marked with 'R' in the 'XXMX\_LOADFILE\_STATUS\_LOG' table and processes each ID sequentially. As it performs operations for each ID, it marks status it as 'i' in log table, indicating that it's in progress.
* The integration checks the 'batch\_load' flag in the 'xxmx\_migration\_metadata' table. If the flag is set to 'Y' (indicating batching), it invokes the 'XXMX\_WRITE\_ORACLEDB\_SCH' procedure, which internally calls the **'XXMX\_WRITE\_DBCS**' procedure from the 'xxmx\_utilities\_pkg' package.
* This procedure extracts line content from the 'xxmx\_csv\_file\_temp' table for finance or 'xxmx\_hdl\_file\_temp' table for HCM, writes the data to the DB file server in batches, and compresses the files using the **'xxmx\_zip\_file**' procedure from the **'xxmx\_utilities\_pkg'** package. The generated files are placed in a specific path provided in the log table.
* Once the file generation is complete, the integration updates the status column for the particular load\_file\_id to 'G' (generated) in the 'XXMX\_LOADFILE\_STATUS\_LOG' table.

3. **Moves File from DB server to FTP:**

* This process remains the same for both batching and non-batching scenarios.
* After the load file is generated, the next step involves transferring it from the DB server to the FTP. This task is accomplished through a scheduled integration in OIC named 'INTDM999017 Moves Load File from DB Server to FTP', which executes every 10 minutes.
* The integration selects load\_file\_ids with a 'G' status from the log table. It iterates through each load\_file\_id, retrieving the files from the DB server using a specific path and downloading the zip files. In the FTP server, if any old files exist for the particular Business Entity or Sub-entity folder path, the integration archives these old files into an archival folder. Subsequently, it uploads the downloaded zip files to the FTP server, deletes the files from the DB server, and updates the status to 'C' in the 'XXMX\_LOADFILE\_STATUS\_LOG' table which means load is completed and zipfiles are available in FTP server in particular business entity/sub entity folder.
* The table below provides a list of tables, status, integrations, and description for each detailing their functionalities during load process.

|  |  |
| --- | --- |
| Key | Description |
| Status 'R' | This status in **xxmx\_loadfile\_status\_log** table means Load data is generated in the temp tables and is ready to generate the file |
| Status 'i' | This status in **xxmx\_loadfile\_status\_log** table means that record is being processed and is in progress. |
| Status 'G' | This status in **xxmx\_loadfile\_status\_log** table means file is generated in DB server. |
| Status 'C' | This status in **xxmx\_loadfile\_status\_log** table means load data file is moved to the FTP server and same can be checked in the ftp folder. |
| INTDM999004 Data Migration Load File Generation V2 | Integration to populate load data in the XXMX\_CSV\_FILE\_TEMP/XXMX\_HDL\_FILE\_TEMP |
| INTDM999015 DM Write Load File to DB Server | Integration to write the Load data populated in database table into DB server as a Zip file. |
| INTDM999017 Moves Load File from DB Server to FTP | Integration to move Load File which is generated in DB server to FileZilla. |
| XXMX\_WRITE\_DBCS | This procedure writes the line content from 'xxmx\_csv\_file\_temp/hdl' table into the DB file server. |
| xxmx\_zip\_file | Procedure which zips the batched files in DB server |
| XXMX\_LOADFILE\_STATUS\_LOG | Table which holds the status of the Load process. |

# 4.2. Non-Batching

## 4.2.1. Extract

The non-batching process mirrors the approach used in the previous version of Maximise, where no batching procedures are invoked in this exception case. In this scenario, the Load\_Batch column for finance and Batch\_Name column for HCM are left null, indicating the absence of created batches. Consequently, files are then visible in a sequential single-file format within the FTP folder.

## 4.2.2. Transform

The Transform approach remains consistent, with data being moved from STG to XFM through transform routines in alignment with the non-batching processes carried out during the transform phase.

## 4.2.3. Load

After the extract and transform stages, the next step is the load process.

1. **Load Process:**

* The process starts by loading data into temporary tables ('xxmx\_csv\_file\_temp' for finance and 'xxmx\_hdl\_file\_temp' for HCM), and then recording this information in the 'XXMX\_LOADFILE\_STATUS\_LOG' table with a Load file ID and status set to 'R'. Subsequently, the data prepared in these temporary tables is ready for file generation on the server.

2. **Load File Generation in DB Server:**

* Load files are generated on the DB server through a scheduled integration in OIC named 'INTDM999015 DM Write Load File to DB Server (1.0)', running every 10 minutes. This integration selects Load file IDs marked with 'R' and iterates through each ID, setting its status to 'i' in the table which means in progress.
* For each load\_file\_id, depending on the batch\_load flag(which should be ‘N’) from the xxmx\_migration\_metadata table for the business entity and sub-entity, the system calls the procedure 'XXMX\_LOADDB\_ORACLEDB\_SCH.' This procedure then invokes 'XXMX\_LOADDB\_FILE' for each sub entities under particular BusinessEntity, writes line content from 'xxmx\_csv\_file\_temp' for finance and 'xxmx\_hdl\_file\_temp' for HCM into a file in Db server.
* Afterwards, it records details such as file name, server path, business entity, and load file ID, and updates the status to 'C' in XXMX\_LIST\_DB\_LOADFILE table once each sub entity file is written inside the DB server
* The integration subsequently checks the status of each file. If it is 'C', then it calls the 'XXMX\_ZIP\_FILE\_NONBATCHING' procedure from 'xxmx\_utilities\_pkg' to zip these files in Db server.
* Finally, updates the status to 'G' in 'XXMX\_LOADFILE\_STATUS\_LOG,' indicating that the file has been generated in the DB server for load\_file\_id.

3. **Moves File from DB server to FTP:**

* The process remains the same for both batching and non-batching scenarios.
* After the load file is generated, the next step involves transferring it from the DB server to the FTP. This task is accomplished through a scheduled integration in OIC named 'INTDM999017 Moves Load File from DB Server to FTP', which executes every 10 minutes.
* The integration selects load\_file\_ids with a 'G' status from the log table. It iterates through each load\_file\_id, retrieving the files from the DB server using a specific path and downloading the zip files. In the FTP server, if any old files exist for the particular Business Entity or Sub-entity folder path, the integration archives these old files into an archival folder. Subsequently, it uploads the downloaded zip files to the FTP server, deletes the files from the DB server, and updates the status to 'C' in the 'XXMX\_LOADFILE\_STATUS\_LOG' table which means load is completed and zipfiles are available in FTP server in particular business entity/sub entity folder.

|  |  |
| --- | --- |
| Key | Description |
| Status 'R' | This status in **xxmx\_loadfile\_status\_log** table means Load data is generated in the temp tables and is ready to generate the file |
| Status 'i' | This status in **xxmx\_loadfile\_status\_log** table means that record is being processed and is in progress. |
| Status 'G' | This status in **xxmx\_loadfile\_status\_log** table means file is generated in DB server. |
| Status 'C' | This status in **xxmx\_loadfile\_status\_log** table means load data file is moved to the FTP server and same can be checked in the ftp folder. |
| INTDM999004 Data Migration Load File Generation V2 | Integration to populate load data in the XXMX\_CSV\_FILE\_TEMP/XXMX\_HDL\_FILE\_TEMP |
| INTDM999015 DM Write Load File to DB Server | Integration to write the Load data populated in database table into DB server as a Zip file. |
| INTDM999017 Moves Load File from DB Server to FTP | Integration to move Load File which is generated in DB server to FileZilla. |
| XXMX\_LIST\_DB\_LOADFILE | This table updates the files present in the directory to the table |
| XXMX\_ZIP\_FILE\_NONBATCHING | Procedure which zips the batched files in DB server |
| XXMX\_LOADFILE\_STATUS\_LOG | Table which holds the status of the Load process. |
| XXMX\_LOADDB\_FILE | This procedure writes the line content from 'xxmx\_csv\_file\_temp/hdl' table into the DB file server. |

# Batching and Non-Batching Snippets

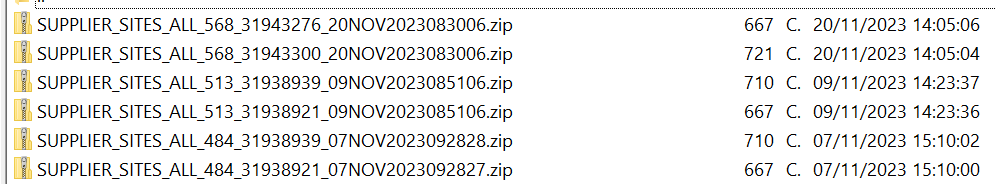
# 1. Batching

* For an example, the files can be observed on the FTP server in the following format,

1. **Business entity level File Generation Example.**

Format: BusinessEntity\_ALL\_BatchName\_LoadFileId\_DateMonthYearTimestamp

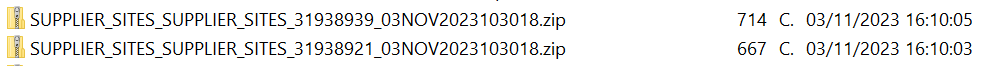
**Example:**

****

**2. SubEntity Level File Generation Example:**

Format: BusinessEntity \_SubEntity\_BatchName\_LoadFileId\_DateMonthYearTimestamp

**Example:**

****

# 2. Non-Batching

* For an example, the files can be observed on the FTP server in the following format,

1. **Business entity level File Generation example.**

Format: BusinessEntity \_ALL\_ LoadFileId \_DateMonthYearTimestamp

Example:



**2. SubEntity Level File Generation Example:**

Format: BusinessEntity \_SubEntity\_ LoadFileId\_DateMonthYearTimestamp







[**www.version1.com**](http://www.version1.com)

*`*