# Teradata to Snowflake Migration Procedure

## Overview

The P\_MIGRATE\_TERADATA\_TO\_SNOWFLAKE procedure is a comprehensive stored procedure that automates the complete DMVA (Data Migration and Validation Automation) process for migrating tables from Teradata to Snowflake. This procedure handles everything from metadata extraction and structure creation to data migration and verification.

## Procedure Signature

CREATE OR REPLACE PROCEDURE NPD\_D12\_DMN\_GDWMIG.MIGRATION\_TRACKING\_V2.P\_MIGRATE\_TERADATA\_TO\_SNOWFLAKE(  
 "P\_SOURCE\_DATABASE\_NAME" VARCHAR, -- Source Teradata database/schema name  
 "P\_SOURCE\_TABLE\_NAME" VARCHAR, -- Source table name  
 "P\_TARGET\_DATABASE\_NAME" VARCHAR, -- Target Snowflake database name  
 "P\_TARGET\_SCHEMA\_NAME" VARCHAR, -- Target Snowflake schema name  
 "P\_TARGET\_TABLE\_NAME" VARCHAR, -- Target table name  
 "P\_WITH\_CHUNKING\_YN" VARCHAR DEFAULT 'N', -- Enable chunking (Y/N)  
 "P\_CHUNKING\_COLUMN" VARCHAR DEFAULT null, -- Column for chunking  
 "P\_CHUNKING\_VALUE" VARCHAR DEFAULT null, -- Chunking value/threshold  
 "P\_CHUNKING\_DATA\_TYPE" VARCHAR DEFAULT null, -- Data type for chunking column  
 "P\_SKIP\_WAITING\_FOR\_MIGRATION\_TASKS" VARCHAR DEFAULT 'N', -- Skip task monitoring (Y/N)  
 "P\_LOAD\_TYPE" VARCHAR DEFAULT 'FULL', -- Load type: FULL or INCREMENTAL  
 "P\_REFRESH\_STRUCTURES\_YN" VARCHAR DEFAULT 'Y' -- Refresh structures (Y/N)  
)  
RETURNS VARCHAR

## Parameters

### Core Migration Parameters

| Parameter | Type | Required | Default | Description |
| --- | --- | --- | --- | --- |
| **P\_SOURCE\_DATABASE\_NAME** | VARCHAR | Yes | - | Source Teradata database or schema name containing the table to migrate |
| **P\_SOURCE\_TABLE\_NAME** | VARCHAR | Yes | - | Name of the source table in Teradata to be migrated |
| **P\_TARGET\_DATABASE\_NAME** | VARCHAR | Yes | - | Target Snowflake database where the migrated table will be created |
| **P\_TARGET\_SCHEMA\_NAME** | VARCHAR | Yes | - | Target Snowflake schema within the target database |
| **P\_TARGET\_TABLE\_NAME** | VARCHAR | Yes | - | Name for the target table in Snowflake (can be different from source) |

### Chunking Parameters

| Parameter | Type | Required | Default | Description |
| --- | --- | --- | --- | --- |
| **P\_WITH\_CHUNKING\_YN** | VARCHAR | No | ‘N’ | Enable chunking for large table migrations. Set to ‘Y’ for tables requiring incremental processing |
| **P\_CHUNKING\_COLUMN** | VARCHAR | No | null | Column name to use for chunking (typically a date or numeric column) |
| **P\_CHUNKING\_VALUE** | VARCHAR | No | null | Threshold value for chunking (e.g., ‘2024-01-01’ for date columns) |
| **P\_CHUNKING\_DATA\_TYPE** | VARCHAR | No | null | Data type of the chunking column (e.g., ‘DATE’, ‘NUMBER’, ‘VARCHAR’) |

### Control Parameters

| Parameter | Type | Required | Default | Description |
| --- | --- | --- | --- | --- |
| **P\_SKIP\_WAITING\_FOR\_MIGRATION\_TASKS** | VARCHAR | No | ‘N’ | Skip monitoring migration tasks. Set to ‘Y’ for asynchronous execution |
| **P\_LOAD\_TYPE** | VARCHAR | No | ‘FULL’ | Load strategy: ‘FULL’ (clean and reload) or ‘INCREMENTAL’ (append data) |
| **P\_REFRESH\_STRUCTURES\_YN** | VARCHAR | No | ‘Y’ | Control structure setup: ‘Y’ (full setup) or ‘N’ (data migration only) |

## Execution Flow

### Complete Migration Flow (P\_REFRESH\_STRUCTURES\_YN = ‘Y’)

#### Phase 1: Structure Setup (Steps 1-4)

**Step 1: Configure Mapping Rules** - Creates or validates mapping rules in dmva\_mapping\_rules table - Links source Teradata objects to target Snowflake objects - Prevents duplicate mappings by checking existing entries - Sets up source system as ‘teradata\_source’ and target as ‘snowflake\_target’

**Step 2: Extract Metadata** - Calls DMVA\_GET\_METADATA\_TASKS to extract table and column metadata from Teradata - Populates dmva\_object\_info and dmva\_column\_info tables - Monitors metadata extraction tasks until completion (max 100 minutes) - Validates metadata was successfully extracted before proceeding

**Step 3: Create Sync Tables** - Calls dmva\_create\_sync\_tables to create target table structures - Establishes column mappings between source and target - Creates necessary indexes and constraints - Populates dmva\_source\_to\_target\_mapping table

**Step 4: Migration Cleanup (Conditional)** - **FULL Load:** Cleans existing checksums and target table data for fresh migration - **INCREMENTAL Load:** Preserves existing data and checksums - Provides row count verification before cleanup - Ensures clean state for data migration

#### Phase 2: Data Migration (Steps 5.1-5.3)

**Step 5.1: Configure Chunking (Optional)** - Validates chunking parameters when enabled - Calls P\_SET\_CHUNKING\_KEY to configure incremental processing - Sets up chunking strategies for large tables - Validates chunking configuration before proceeding

**Step 5.2: Data Migration** - Calls DMVA\_GET\_CHECKSUM\_TASKS to initiate data transfer - Handles data type conversions and transformations - Manages parallel processing for optimal performance - Creates checksums for data validation

**Step 5.3: Monitor Migration Tasks** - Monitors migration task progress (max 200 minutes) - Provides real-time status updates - Waits for all tasks to complete successfully - Can be skipped for asynchronous processing

### Data-Only Migration Flow (P\_REFRESH\_STRUCTURES\_YN = ‘N’)

When structures already exist, the procedure executes only the data migration phase (Steps 5.1-5.3), assuming: - Mapping rules are already configured - Metadata has been extracted - Target structures exist - Source-to-target mappings are established

## Usage Examples

### Example 1: Complete Fresh Migration

-- Full migration with structure setup and data cleanup  
CALL NPD\_D12\_DMN\_GDWMIG.MIGRATION\_TRACKING\_V2.P\_MIGRATE\_TERADATA\_TO\_SNOWFLAKE(  
 'K\_PDDSTG', -- Source Teradata schema  
 'CUSTOMER\_DATA', -- Source table  
 'NPD\_D12\_DMN\_GDWMIG', -- Target database  
 'MIGRATION', -- Target schema  
 'TD\_CUSTOMER\_DATA', -- Target table name  
 'N', -- No chunking  
 null, null, null, -- Chunking parameters (not used)  
 'N', -- Wait for completion  
 'FULL', -- Fresh migration  
 'Y' -- Full structure setup  
);

### Example 2: Large Table Migration with Date Chunking

-- Migration of large table using date-based chunking  
CALL NPD\_D12\_DMN\_GDWMIG.MIGRATION\_TRACKING\_V2.P\_MIGRATE\_TERADATA\_TO\_SNOWFLAKE(  
 'K\_PDDSTG', -- Source schema  
 'TRANSACTION\_HISTORY', -- Large source table  
 'NPD\_D12\_DMN\_GDWMIG', -- Target database  
 'STAGING', -- Target schema  
 'TD\_TRANSACTION\_HISTORY', -- Target table  
 'Y', -- Enable chunking  
 'TRANSACTION\_DATE', -- Date column for chunking  
 '2024-01-01', -- Start from this date  
 'DATE', -- Date data type  
 'N', -- Monitor tasks  
 'FULL', -- Fresh load  
 'Y' -- Setup structures  
);

### Example 3: Incremental Data Load

-- Incremental load preserving existing data  
CALL NPD\_D12\_DMN\_GDWMIG.MIGRATION\_TRACKING\_V2.P\_MIGRATE\_TERADATA\_TO\_SNOWFLAKE(  
 'K\_PDDSTG', -- Source schema  
 'DAILY\_UPDATES', -- Source table  
 'NPD\_D12\_DMN\_GDWMIG', -- Target database  
 'INCREMENTAL', -- Target schema  
 'TD\_DAILY\_UPDATES', -- Target table  
 'N', -- No chunking needed  
 null, null, null, -- Chunking parameters  
 'N', -- Wait for completion  
 'INCREMENTAL', -- Preserve existing data  
 'Y' -- Full setup  
);

### Example 4: Data-Only Migration (Structures Exist)

-- Data migration only, assuming structures already exist  
CALL NPD\_D12\_DMN\_GDWMIG.MIGRATION\_TRACKING\_V2.P\_MIGRATE\_TERADATA\_TO\_SNOWFLAKE(  
 'K\_PDDSTG', -- Source schema  
 'REFERENCE\_DATA', -- Source table  
 'NPD\_D12\_DMN\_GDWMIG', -- Target database  
 'REFERENCE', -- Target schema  
 'TD\_REFERENCE\_DATA', -- Target table  
 'N', -- No chunking  
 null, null, null, -- Chunking parameters  
 'N', -- Monitor completion  
 'FULL', -- Clean load  
 'N' -- Skip structure setup  
);

### Example 5: Asynchronous Migration

-- Fire-and-forget migration without waiting for completion  
CALL NPD\_D12\_DMN\_GDWMIG.MIGRATION\_TRACKING\_V2.P\_MIGRATE\_TERADATA\_TO\_SNOWFLAKE(  
 'K\_PDDSTG', -- Source schema  
 'ASYNC\_TABLE', -- Source table  
 'NPD\_D12\_DMN\_GDWMIG', -- Target database  
 'ASYNC', -- Target schema  
 'TD\_ASYNC\_TABLE', -- Target table  
 'N', -- No chunking  
 null, null, null, -- Chunking parameters  
 'Y', -- Skip waiting (async)  
 'FULL', -- Fresh migration  
 'Y' -- Full setup  
);

### Example 6: Numeric Chunking for Large Tables

-- Migration using numeric column for chunking  
CALL NPD\_D12\_DMN\_GDWMIG.MIGRATION\_TRACKING\_V2.P\_MIGRATE\_TERADATA\_TO\_SNOWFLAKE(  
 'K\_PDDSTG', -- Source schema  
 'LARGE\_NUMERIC\_TABLE', -- Source table  
 'NPD\_D12\_DMN\_GDWMIG', -- Target database  
 'CHUNKED', -- Target schema  
 'TD\_LARGE\_NUMERIC\_TABLE', -- Target table  
 'Y', -- Enable chunking  
 'ACCOUNT\_ID', -- Numeric column for chunking  
 '1000000', -- Chunk at 1M account IDs  
 'NUMBER', -- Numeric data type  
 'N', -- Monitor completion  
 'FULL', -- Fresh migration  
 'Y' -- Full setup  
);

## Return Value

The procedure returns a detailed execution log containing:

* **Timestamps:** Each step execution time
* **Parameter Values:** All input parameters and their validation
* **Progress Updates:** Real-time status of metadata and migration tasks
* **Row Counts:** Before and after migration verification
* **Error Messages:** Detailed error information if failures occur
* **Loop Counters:** Task monitoring progress indicators
* **Final Status:** Migration completion confirmation

## Error Handling and Validation

### Parameter Validation

* **P\_LOAD\_TYPE:** Must be ‘FULL’ or ‘INCREMENTAL’
* **P\_REFRESH\_STRUCTURES\_YN:** Must be ‘Y’ or ‘N’
* **P\_SKIP\_WAITING\_FOR\_MIGRATION\_TASKS:** Must be ‘Y’ or ‘N’
* **Chunking Parameters:** When chunking is enabled, all chunking parameters must be provided

### Runtime Validation

* **Metadata Verification:** Ensures source table metadata was extracted successfully
* **Structure Validation:** Verifies target structures and mappings were created
* **Chunking Validation:** Validates chunking configuration before data migration
* **Task Monitoring:** Tracks task completion with timeout protection

### Exception Handling

* **Comprehensive Error Logging:** All errors are captured with timestamps
* **Graceful Degradation:** Procedure continues where possible or fails safely
* **Detailed Error Messages:** Specific error context for troubleshooting
* **Parameter Context:** Error messages include problematic parameter values

## Dependencies

### Required Procedures

* **DMVA\_GET\_METADATA\_TASKS:** Extracts metadata from Teradata source
* **dmva\_create\_sync\_tables:** Creates target structures and mappings
* **DMVA\_GET\_CHECKSUM\_TASKS:** Performs data migration and validation
* **P\_SET\_CHUNKING\_KEY:** Configures chunking parameters (when chunking enabled)

### Required Tables

* **dmva\_mapping\_rules:** Source-to-target object mappings
* **dmva\_object\_info:** Object metadata (tables, views)
* **dmva\_column\_info:** Column metadata and data types
* **dmva\_source\_to\_target\_mapping:** Column-level mappings
* **dmva\_tasks:** Task execution tracking
* **dmva\_checksums:** Data validation checksums

### System Dependencies

* **Sequences:** dmva\_mapping\_rule\_id\_seq for unique mapping IDs
* **System Functions:** SYSTEM$WAIT for task monitoring delays
* **Dynamic SQL:** For runtime target table operations

## Monitoring and Performance

### Built-in Monitoring

* **Metadata Tasks:** Real-time monitoring with 2-second intervals (max 100 minutes)
* **Migration Tasks:** Progress tracking with 2-second intervals (max 200 minutes)
* **Row Count Verification:** Before and after migration counts
* **Task Status Tracking:** Running, completed, and failed task identification

### Performance Considerations

* **Chunking:** Use for tables > 10M rows or complex data types
* **Parallel Processing:** DMVA framework handles parallel task execution
* **Timeout Protection:** Prevents infinite loops with configurable limits
* **Asynchronous Option:** Skip monitoring for long-running migrations

### Troubleshooting

* **Detailed Logging:** Every step logged with timestamps
* **Parameter Echo:** All input parameters logged for debugging
* **Error Context:** Specific failure points identified
* **Progress Indicators:** Loop counters and task counts for monitoring

## Best Practices

### When to Use Full vs Incremental

* **FULL:** New migrations, data corrections, complete refreshes
* **INCREMENTAL:** Regular updates, append-only scenarios, preserving historical data

### Chunking Strategies

* **Date Columns:** Use for time-series data with natural chronological boundaries
* **Numeric IDs:** Use for large tables with sequential or distributed numeric keys
* **No Chunking:** Small tables (< 1M rows) or simple data types

### Structure Refresh Guidelines

* **Y (Full Setup):** New migrations, schema changes, mapping updates
* **N (Data Only):** Regular data refreshes, scheduled updates, established pipelines

### Monitoring Approach

* **Synchronous:** For critical migrations requiring immediate validation
* **Asynchronous:** For large migrations during off-peak hours
* **Hybrid:** Start synchronous, switch to asynchronous for long-running tasks