Zamboni Detailed Design

Version 1

Marc Henderson

April 29, 2024

Revision Summary

Date	Revision History	Comments
04/29/2024	1.0	Initial Version
10/17/2024	1.1	Added new/updated functionality and documented metadata JSON schemas

Table of Contents

Terminology

- **Object** - a table, dynamic table, view, materialized, or file (coming soon) that contains data to be included in a collection. Objects can exist in multiple collections.
- **Collection** - a grouping of objects and applicable columns/fields to either map to another object in target collection or be mapped to from a source collection. A collection can contain multiple objects.
  - **source collection** - the collection containing objects and their columns/fields to update objects in a target collection.
  - **target collection** - the collection containing objects to be mapped to, from objects in a source collection.
- **Process** - a mapping between objects in source and target collections. A process defines the parameters to either create a new target object (i.e. dynamic table or view) or incrementally update an existing object (i.e. a table).
- **Process Type** - the type of operation to be performed, based on the process mapping. For example, creating a dynamic table, or merging into an existing table.
  - Process types are defined as Jinja templates that are used to generate the DDL/DML statements, using the process mapping.
  - Process Types:
    - target_dynamic_table
    - target_incremental_merge_insert
    - target_standard_table
    - target_standard_view
    - target_materialized_view
    - target_incremental_merge_delete
    - target_file (coming soon)
- **DAG** - a directed acyclic graph that defines a parent process and orchestrates child processes to be executed in a predefined order. A DAG must have at least one child process.
  - Using Snowflake nomenclature, DAGs contain one or more tasks that execute commands that create/update objects defined in the process mapping.
- **Label** - a unique descriptor for any object, collection, process, or DAG. Labels are optional, but allows for end-to-end traceability based on a label or group of labels.

Metadata

Metadata for Zamboni collections, process mappings, and DAGs are stored as a JSON object (VARIANT). JSON allows the ability to easily extend the attributes collected for each metadata type, as needed.

The following sections describe the JSON schemas for each Zamboni metadata type. Refer to the Appendix for examples

Collections

See Sample JSON Collections Payload for an example.

- **objects** (ARRAY) - an array of one or more objects and their columns that belong to the collection.
    - **alias** (STRING) - the object's alias used in the SELECT statement.
    - **columns** (ARRAY) - an array of one or more of the desired columns.
    - **object_id** (INTEGER) - the unique ID for the object, as stored in the OBJECTS table.

Processes

See Sample Create Process: Create Dynamic Table JSON Payload and Sample Create Process: Incremental Merge/Insert JSON Payload for examples.

- **process_name** (STRING) - a unique name for the process mapping.
- **process_type_id** (STRING) - the unique ID for the process_type, as stored in the PROCESS_TYPES table. This ID will be used to get the appropriate template to build the DDL/DML statement.
- **target** (OBJECT) - the target object the process should create/update.
    - **collection_id** (INTEGER) - if the target object exists, the ID of the collection, as stored in the COLLECTIONS table. If the target object is to be created, it will be added to the collection assigned to this ID.
    - **alias** (STRING) - the object's alias used in SELECT statements. If the target object is to be created, the alias should be unique and will be assigned to the object, when added to the specified collection.
    - **object** (STRING) - the fully-qualified name of the object to either create or update.
- **source** (OBJECT) - the SELECT statement's source object. This is the object immediately after FROM.

    - **collection_id** (INTEGER) - the ID of the collection the source object belongs to, as stored in the COLLECTIONS table.
    - **alias** (STRING) - the object's alias used in SQL statements.
    - **key** (STRING) - the column/value to join other tables on.
    - **object** (STRING) - the fully-qualified name of the source object.
- **distinct** (BOOLEAN) - flag to determine whether the process' SELECT statement should include DISTINCT.
- **top** (INTEGER) - flag to determine whether the process' SELECT statement should include TOP n number of records. If TOP is not required, this should be set to **null**.
- **columns** (ARRAY) - an array of one or more of the desired columns from the source collection(s) to include in the process' SELECT statement. Columns can include functions and scalar values. Object aliases should be included.
- **where** (ARRAY) - an array of zero or more clauses to filter the process' SELECT statement.
    - **attr_1** (STRING) - the left side of the WHERE clause. The object's alias should be included.
    - **operator** (STRING) - the operator used to compare the attr_1 and attr_2 sides of the WHERE clause.
    - **attr_2** (STRING) the right side of the WHERE clause.
    - **condition** (STRING) - the condition used to compare multiple WHERE clauses, if applicable. If there is either one clause or this is the last clause, this value is an empty string.
- **join** (ARRAY) - an array of zero or more objects to join to the source object, if applicable.
    - **collection_id** (INTEGER) - the ID of the collection the source object belongs to, as stored in the COLLECTIONS table.
    - **alias** (STRING) - the object's alias used in SELECT statements.
    - **object** (STRING) - the fully-qualified name of the source object.
    - **keys** (ARRAY) - an array of one or more JOIN conditions used to join the source objects.
        - **attr_1** (STRING) - the column from source or other join table (if joining multiple tables) to join on. The object's alias should be included.
        - **operator** (STRING) - the equals sign.
        - **attr_2** (STRING) the column from this table to join on. The object's alias should be included.
- **group_by** (ARRAY) - an array of zero or more of the desired columns/functions/values to group by, if applicable.
- **having** (ARRAY) - an array of zero or more clauses to filter the rows produced by the GROUP BY clause(s).
    - **attr_1** (STRING) - the constant, GROUP BY expression, or aggregate function.
    - **operator** (STRING) - the operator used to compare the attr_1 and attr_2 sides of the HAVING clause.
    - **attr_2** (STRING) the value to compare attr_1 to

- **order_by_cols** (ARRAY) - an array of zero or more of the desired columns/functions/values to order by, if applicable. If this is used, **do not** use order_by_pos.
- **order_by_pos** (ARRAY) - an array of zero or more numbers, specifying the position of the column(s) in the SELECT statement to order by. If this is used, **do not** use order_by_cols.
- **mapping** (ARRAY) - an array of zero or more source-to-target mappings to either merge on, insert and/or update. This is only applicable for MERGE process types.
    - **source_attr** (STRING) - the column from the source object
    - **target_attr** (STRING) - the column from the target object
    - **merge_on** (STRING) -Y/N flag indicating whether to use mapping to merge on.
    - **update** (STRING) - Y/N flag indicating whether to update the target_attr to the source_attr value, when the merge_on mapping matches.
    - **insert** (STRING) - Y/N flag indicating whether to insert the source_attr value into the target table, when the merge_on mapping does not match.
- **labels** (ARRAY) - an array of zero or more labels, as defined in the LABELS table, to assign to the process mapping

Process DAG

See Sample Manage Process DAG JSON Payload for an example.

- **child_processes** (ARRAY) - an array of one or more processes to include in the DAG and the order in which the process should be created/executed.
    - **process_id** (INTEGER) - the unique ID of the process mapping, as defined in the PROCESSES table.
    - **process_name** (STRING) - the name of the process mapping, as defined in the PROCESSES table.
    - **process_order** (INTEGER) - the order in which the process should be executed in the DAG.

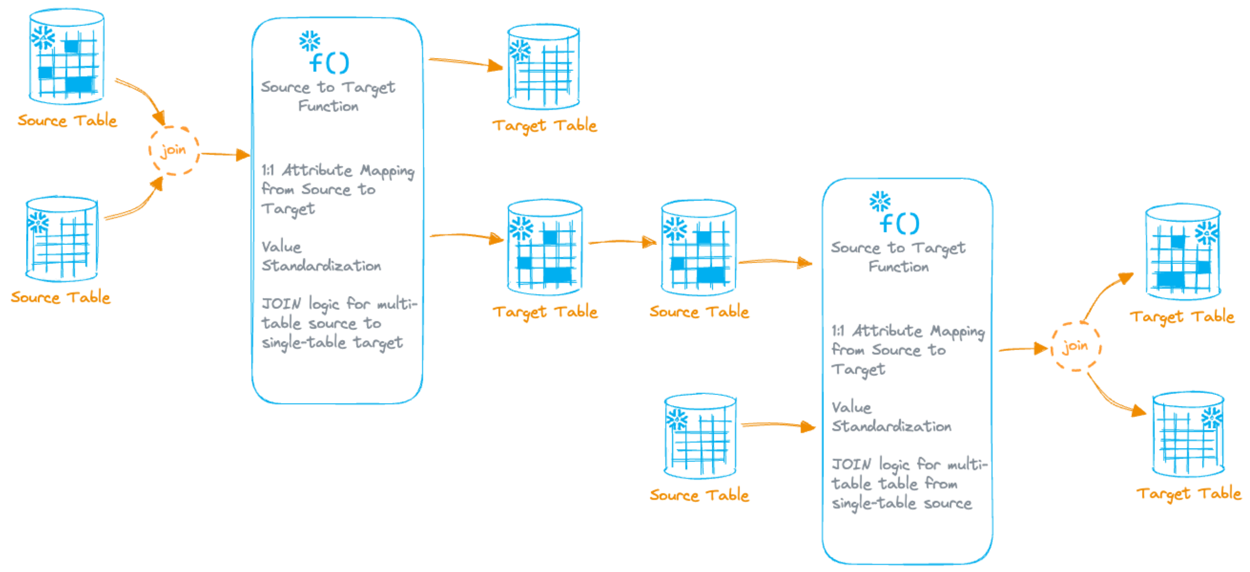
Detailed Design

The Zamboni framework will allow a customer to create a pipeline of Source-to-Target transformations with little to no ELT engineering required. The framework will be built to allow customers to choose a source and map attributes to a specific target layout, including simple data transformations.

The Zamboni framework will include the following major components:

- Source to Target transformation module
- An orchestration management module to string together multiple source-to-targets
- A UI to define sources, targets, and transformations (coming soon)
- A series of operational reports, including PF/FK constraint checks, field value trends, error record handling, etc.

Design Diagram


Source Table
join
Source Table
f()
Source to Target Function
1:1 Attribute Mapping from Source to Target
Value Standardization
JOIN logic for multi-table source to single-table target
Target Table
Target Table
Source Table
Source Table
f()
Source to Target Function
1:1 Attribute Mapping from Source to Target
Value Standardization
JOIN logic for multi-table table from single-table source
join
Target Table
Target Table

Objects Created

Primary Account

The following objects are created when the Zamboni scripts are executed.

Database: ZAMBONI_DB

Description
The database that stores the Zamboni objects.

Schema: ZAMBONI_METADATA

Description:
This schema stores the metadata objects.

Table: LABELS

Description
Table containing the labels applied to objects, collections, processes, and process DAGs.

Definition

Column	Data Type	Description	Null?
LABEL_ID	INT	The label's unique identifier (Primary Key).	N
LABEL_NAME	VARCHAR	Any descriptive text.	N
DESCRIPTION	VARCHAR	The description of the label.	N
ATTRIBUTES	VARIANT	Attributes about the label. **NOTE**: this is an optional field that can be used to store any additional information about the label.	Y
CREATED_TIMESTAMP	TIMESTAMP_NTZ	The timestamp when the label was created.	N
MODIFIED_TIMESTAMP	TIMESTAMP_NTZ	The timestamp when the label was modified.	Y

Table: OBJECTS

Description
Table containing the metadata about the objects in a given collection.

Definition

Column	Data Type	Description	Null?
OBJECT_ID	INT	The object's unique identifier (Primary Key).	N
OBJECT_TYPE	VARCHAR	The type of object. Object types include: ● **table** ● **dynamic_table** ● **view** ● **materialized_view** ● **file (coming soon)**	N
DATABASE_NAME	VARCHAR	The database where the object resides.	N
SCHEMA_NAME	VARCHAR	The schema where the object resides.	N
OBJECT_NAME	VARCHAR	The name of the object.	N
ATTRIBUTES	VARIANT	Object attributes, in JSON format. The attributes depend on the object type.	N
LABELS	ARRAY	An array of labels (descriptors), applicable to the object.	Y
ADDED_TIMESTAMP	TIMESTAMP_NTZ	The timestamp when the object was added.	N
MODIFIED_TIMESTAMP	TIMESTAMP_NTZ	The timestamp when the object was modified.	Y

Table: COLLECTIONS

Description
Table containing the metadata about a given collection.

Definition

Column	Data Type	Description	Null?
COLLECTION_ID	INT	The collection's unique identifier (Primary Key).	N
COLLECTION_NAME	VARCHAR	The name of the collection.	N
COLLECTION_TYPE	VARCHAR	The type of collection. Collection types:	

		• **standard** • **custom** (coming soon)	
OBJECTS	VARIANT	The objects and their columns that are included in the collection, in JSON format.	N
LABELS	ARRAY	An array of labels (descriptors), applicable to the collection.	Y
CREATED_TIMESTAMP	TIMESTAMP_NTZ	The timestamp when the collection was created.	N
MODIFIED_TIMESTAMP	TIMESTAMP_NTZ	The timestamp when the collection was modified.	Y

Table: PROCESS_TYPES

Description
Table containing the process type definitions.

Definition

Column	Data Type	Description	Null?
PROCESS_TYPE_ID	INT	The process' unique identifier (Primary Key).	N
PROCESS_TYPE	VARCHAR	The process type.	N
DESCRIPTION	VARCHAR	The description of the process type.	Y
TEMPLATE	VARIANT	The jinja template, used to construct the SQL statement to map source columns to target columns.	N
OBJECT_TYPE	VARCHAR	The type of target object the process type creates/updates Types include: • **table** • **dynamic_table** • **view** • **materialized_view** • **file** (coming soon)	N
OBJECT_ACTION	VARCHAR	The action the process type performs: The actions include: • **create** • **merge_insert** • **merge_delete**	N
CREATED_TIMESTAMP	TIMESTAMP_NTZ	The timestamp when the process type was created.	N

MODIFIED_TIMESTAMP	TIMESTAMP_NTZ	The timestamp when the process type was modified.	Y

Table: PROCESSES

Description
Table containing the metadata about a given process.

Definition

Column	Data Type	Description	Null?
PROCESS_ID	INT	The process' unique identifier (Primary Key).	N
PROCESS_NAME	VARCHAR	The name of the process.	N
PROCESS_TYPE_ID	INT	The identifier for the type of process (Foreign Key).	N
ATTRIBUTES	VARIANT	The source/target collection(s), the mappings between columns in the source collection to the columns in the target collection, and any transformations, in JSON format. **NOTE**: The attributes are used to construct the SQL command to create/update the target object	N
LABELS	ARRAY	An array of labels (descriptors), applicable to the process.	Y
CREATED_TIMESTAMP	TIMESTAMP_NTZ	The timestamp when the process was created.	N
MODIFIED_TIMESTAMP	TIMESTAMP_NTZ	The timestamp when the process was modified.	Y

Table: PROCESS_DAG

Description
Table containing the process DAG definitions.

Definition

Column	Data Type	Description	Null?
PARENT_PROCESS_ID	INT	The parent process' unique identifier (Primary Key).	N
PARENT_PROCESS_NAME	VARCHAR	The name of the parent process/dag.	N

CHILD_PROCESSES	VARIANT	The child processes associated with the DAG and their execution order, in JSON format. The CREATE_PROCESS stored procedure executes the defined child processes, in the order specified.	N
LABELS	ARRAY	An array of labels (descriptors), applicable to the process DAG.	Y
CREATED_TIMESTAMP	TIMESTAMP_NTZ	The timestamp when the process DAG was created.	N
MODIFIED_TIMESTAMP	TIMESTAMP_NTZ	The timestamp when the process DAG was modified.	Y

Table: PROCESS_LOG

Description
Table containing the log of the process DAG executions.

Definition

Column	Data Type	Description	Null?
PROCESS_LOG_ID	INT	The log entry's unique identifier (Primary Key).	N
PROCESS_RUN_ID	INT	The identifier of the process' execution.	N
PARENT_PROCESS_ID	INT	The identifier of the process' parent process (Foreign Key).	N
PROCESS_ID	INT	The process' unique identifier	N
PROCESS_START_TIMESTAMP	TIMESTAMP_NTZ	The timestamp when the process run started.	N
PROCESS_END_TIMESTAMP	TIMESTAMP_NTZ	The timestamp when the process run ended.	Y
PROCESS_OUTPUT	VARIANT	The output of the process runs, including any pass/fail messages, in JSON format.	N

Schema: ZAMBONI_UTIL

Description:

This schema creates the functions and procedures leveraged by Zamboni to create and manage various components.

Stored Procedure: MANAGE_LABEL

Description:
This stored procedure adds/updates labels to be used to attach to objects, collections, processes, and process DAGs. This procedure should be called either when consumers are adding labels or when attempting to add labels that do not exist in the LABELS table.

Parameters:
- **label_name** (VARCHAR) - the name of the label.
- **description** (VARCHAR) - the description of the label.
- **attributes** (VARIANT) - Attributes about the label. **NOTE**: this is an optional field that can be used to store additional information about the label.

Sample Call:

Unset
```
CALL MANAGE_LABEL('label1', 'A label to organize related objects, collections,
and processes/dags for: label1', NULL);
```

Stored Procedure: MANAGE_OBJECT

Description:
This stored procedure adds/updates an object that belongs to a collection, to the OBJECTS table. This procedure should be called when the consumer selects an object to be included in the collection.

Parameters:
- **object_type** (VARCHAR) - the type of object being added/updated in a collection.
- **database_name** (VARCHAR) - the database where the object resides.
- **schema_name** (VARCHAR) - the schema where the object resides.
- **object_name** (VARCHAR) - the object name.
- **stage_location** (VARCHAR) - the stage where the file is located, when object_type = file (coming soon).
- **labels** (ARRAY) - any descriptors that are applicable to the object.

Sample Call:

Unset
```
CALL MANAGE_OBJECT('table', 'ZAMBONI_DB', 'ZAMBONI_SRC', 'INVENTORY_ON_HANDS',
NULL, ARRAY_CONSTRUCT('label1','label2'));
```

Stored Procedure: MANAGE_COLLECTION

Description:
This stored procedure adds/updates a collection, including the included objects and columns, to the COLLECTIONS table. This procedure should be called after the consumer selects objects and columns to include in the collection.

Parameters:
- **objects** (VARIANT) - the objects and their columns that are included in the collection, in JSON format.
- **collection_name** (VARCHAR) - the name of the collection.
- **prev_collection_name** (VARCHAR) - the previous name of the collection (if updating an existing collection).
- **collection_type** (VARCHAR) - the type of collection: **standard** and **custom** (coming soon)
- **labels** (ARRAY) - any descriptors that are applicable to the collection.

Sample Call:

Unset
```
CALL MANAGE_COLLECTION(PARSE_JSON($$
{
    "objects": [
      {
        "object_id" : 1,
        "attributes" : [
          "ITEMID",
          "LOCATIONID",
          "PROJECT",
          "TYPE",
          "AVAILABLEFORSUPPLYDATE",
          "BATCH",
          "UOM",
          "QUANTITY",
          "PROCESSTYPE",
          "EXPIRATIONDATE",
          "SITEOWNER",
          "ITEMOWNER",
          "ONHANDPOSTDATETIME",
          "MEASURE",
          "NODETYPE",
          "LOB",
          "LOTNUMBER",
          "STORE",
          "CTOITEMID",
          "CTOBOMID"
        ]
      },
      {
        "object_id" : 2,
        "attributes" : [
          "STARTDATE",
          "TRANSACTIONCODE"
        ]
      }
    ]
}$$), 'COL_INVENTORY_TRANSACTION_1', NULL, 'standard',
ARRAY_CONSTRUCT('label1','label2'));
```

Stored Procedure: MANAGE_PROCESS

Description:
For each process passed to it, this stored procedure adds/updates the process to the PROCESSES table, including the source to target mappings for each process. This procedure should be called after the consumer creates mappings between the columns in source and target collections.

Parameters:

- **processes** (VARIANT) - the variant containing one or more source-to-target mappings.

Sample Call:

Unset
```
CALL MANAGE_PROCESS(PARSE_JSON($$
{
  "targets" : [
    {
      "process_name" : "target_dt_inventory_by_transaction",
      "process_type_id" : 1,
      "distinct": true,
      "top": null,
      "columns" : [
        "OBJ_1.ITEMID",
        "OBJ_1.LOCATIONID",
        "MAX(OBJ_1.PROJECT) PROJECT",
        "MAX(OBJ_1.TYPE) TYPE",
        "MAX(OBJ_1.AVAILABLEFORSUPPLYDATE) AVAILABLEFORSUPPLYDATE",
        "RIGHT(OBJ_1.BATCH, 4) BATCH",
        "SUM(OBJ_1.QUANTITY) QUANTITY_SUM",
        "MAX(OBJ_1.STORE) STORE",
        "MAX(OBJ_2.STARTDATE) STARTDATE",
        "MAX(OBJ_2.TRANSACTIONCODE) TRANSACTIONCODE"
      ],
      "group_by" : [
        "OBJ_1.ITEMID",
      "OBJ_1.LOCATIONID",
      "RIGHT(OBJ_1.BATCH, 4)"
    ],
    "join" : [
      {
        "collection_id" : 1,
        "alias" : "OBJ_2",
        "object" : "ZAMBONI_DB.ZAMBONI_SRC.INVENTORY_TRANSACTIONS",
        "keys" : [
          {
            "attr_1" : "OBJ_1.ITEMID",
            "operator" : "=",
            "attr_2" : "OBJ_2.ITEMID",
            "condition" : ""
          }
        ]
      }
    ],
    "order_by_cols" : [
      "ITEMID",
      "LOCATIONID",
      "BATCH",
      "AVAILABLEFORSUPPLYDATE"
    ],
    "settings" : {
      "downstream" : true,
      "target_interval" : "hours",
      "target_lag" : 24,
      "warehouse" : "xs_wh"
    },
    "source" : {
      "collection_id" : 1,
    "alias" : "OBJ_1",
    "key" : "ITEMID",
    "object" : "ZAMBONI_DB.ZAMBONI_SRC.INVENTORY_ON_HANDS"
  },
  "target" : {
    "collection_id" : 2,
    "alias" : "OBJ3",
    "key" : null,
    "object" : "ZAMBONI_DB.ZAMBONI_TGT.INVENTORY_BY_TRANSACTION_DYNAMIC"
  },
  "where" : [
    {
      "attr_1" : "OBJ_1.LOCATIONID",
      "operator" : "IN",
      "attr_2" : "('LDC2', 'LDC3')",
      "condition" : "AND"
    },
    {
      "attr_1" : "OBJ_1.ITEMID",
      "operator" : "!=",
      "attr_2" : "'CFGB09'",
      "condition" : "AND"
    },
    {
      "attr_1" : "OBJ_1.AVAILABLEFORSUPPLYDATE ",
      "operator" : "!=",
      "attr_2" : "'2023-10-30'",
      "condition" : ""
    }
  ],
  "mapping" : [],
  "labels" : ["label1", "label2"]
```

```
    },
    {
      "process_name" : "target_incremental_merge_inventory_by_transaction_1",
      "process_type_id" : 2,
      "distinct": false,
      "top": 1000,
      "columns" : [

"HASH(OBJ_1.ITEMID,OBJ_1.LOCATIONID,MAX(OBJ_1.PROJECT),MAX(OBJ_1.AVAILABLEFORSU
PPLYDATE),RIGHT(OBJ_1.BATCH,
4),SUM(OBJ_1.QUANTITY),MAX(OBJ_1.STORE),MAX(OBJ_2.STARTDATE),MAX(OBJ_2.TRANSACT
IONCODE)) RECORD_ID" ,
        "OBJ_1.ITEMID",
        "OBJ_1.LOCATIONID",
        "MAX(OBJ_1.PROJECT) PROJECT",
        "MAX(OBJ_1.TYPE) TYPE",
        "MAX(OBJ_1.AVAILABLEFORSUPPLYDATE) AVAILABLEFORSUPPLYDATE",
        "RIGHT(OBJ_1.BATCH, 4) BATCH",
        "SUM(OBJ_1.QUANTITY) QUANTITY_SUM",
        "MAX(OBJ_1.STORE) STORE",
        "MAX(OBJ_2.STARTDATE) STARTDATE",
        "MAX(OBJ_2.TRANSACTIONCODE) TRANSACTIONCODE"
      ],
      "group_by" : [
        "OBJ_1.ITEMID",
        "OBJ_1.LOCATIONID",
        "RIGHT(OBJ_1.BATCH, 4)"
      ],
      "join" : [
        {
          "collection_id" : 1,
          "alias" : "OBJ_2",
          "key" : "ITEMID",
          "object" : "ZAMBONI_DB.ZAMBONI_SRC.INVENTORY_TRANSACTIONS"
        }
      ],
      "order_by_pos" : [1,2,3,7,6],
      "settings" : {
        "target_interval" : "minute",
        "target_lag" : 1440,
        "warehouse" : "xs_wh",
        "when_matched" : [],
        "when_not_matched" : []
      },
      "source" : {
        "collection_id" : 1,
        "alias" : "OBJ_1",
        "key" : "ITEMID",
        "object" : "ZAMBONI_DB.ZAMBONI_SRC.INVENTORY_ON_HANDS"
      },
      "target" : {
        "collection_id" : 2,
        "alias" : "TGT_1",
        "key" : null,
        "object" :
"ZAMBONI_DB.ZAMBONI_TGT.INVENTORY_BY_TRANSACTION_INCREMENTAL"
      },
      "where" : [
        {
          "attr_1" : "OBJ_1.LOCATIONID",
          "operator" : "IN",
          "attr_2" : "('LDC2', 'LDC3')",
          "condition" : "AND"
        },
        {
          "attr_1" : "OBJ_1.ITEMID",
          "operator" : "!=",
          "attr_2" : "'CFGB09'",
          "condition" : "AND"
        },
        {
          "attr_1" : "OBJ_1.AVAILABLEFORSUPPLYDATE ",
          "operator" : "!=",
          "attr_2" : "'2023-10-30'",
          "condition" : ""
        }
      ],
      "mapping" : [
        {
            "source_attr" : "RECORD_ID",
            "target_attr" : "RECORD_ID",
            "merge_on" : "Y",
            "update" : "N",
            "insert" : "Y"
        },
        {
            "source_attr" : "ITEMID",
            "target_attr" : "ITEM_ID",
            "merge_on" : "N",
            "update" : "Y",
            "insert" : "Y"
        },
        {
            "source_attr" : "LOCATIONID",
            "target_attr" : "LOCATION_ID",
            "merge_on" : "N",
            "update" : "Y",
            "insert" : "Y"
        },
        {
            "source_attr" : "PROJECT",
            "target_attr" : "PROJECT_NAME",
            "merge_on" : "N",
            "update" : "Y",
            "insert" : "Y"
        },
        {
          "source_attr" : "TYPE",
          "target_attr" : "TYPE",
          "merge_on" : "N",
          "update" : "Y",
          "insert" : "Y"
        },
        {
          "source_attr" : "AVAILABLEFORSUPPLYDATE",
          "target_attr" : "SUPPLY_DATE",
          "merge_on" : "N",
          "update" : "Y",
          "insert" : "Y"
        },
        {
          "source_attr" : "BATCH",
          "target_attr" : "BATCH_ID",
          "merge_on" : "N",
          "update" : "Y",
          "insert" : "Y"
        },
        {
          "source_attr" : "QUANTITY_SUM",
          "target_attr" : "QUANTITY_SUM",
          "merge_on" : "N",
          "update" : "Y",
          "insert" : "Y"
        },
        {
          "source_attr" : "STORE",
          "target_attr" : "STORE_NAME",
          "merge_on" : "N",
          "update" : "Y",
          "insert" : "Y"
        },
        {
          "source_attr" : "STARTDATE",
          "target_attr" : "START_DATE",
          "merge_on" : "N",
          "update" : "Y",
          "insert" : "Y"
        },
        {
          "source_attr" : "TRANSACTIONCODE",
          "target_attr" : "TRANSACTION_CODE",
          "merge_on" : "N",
          "update" : "Y",
          "insert" : "Y"
        }
      ],
      "labels" : ["label1", "label2"]
    },
    {
      "process_name" : "target_incremental_merge_inventory_by_transaction_2",
      "process_type_id" : 2,
      "distinct": false,
      "top": 1000,
      "columns" : [

"HASH(OBJ_1.ITEMID,OBJ_1.LOCATIONID,MAX(OBJ_1.PROJECT),MAX(OBJ_1.AVAILABLEFORSU
PPLYDATE),RIGHT(OBJ_1.BATCH,
4),SUM(OBJ_1.QUANTITY),MAX(OBJ_1.STORE),MAX(OBJ_2.STARTDATE),MAX(OBJ_2.TRANSACT
IONCODE)) RECORD_ID" ,
        "OBJ_1.ITEMID",
        "OBJ_1.LOCATIONID",
        "MAX(OBJ_1.PROJECT) PROJECT",
        "MAX(OBJ_1.TYPE) TYPE",
        "MAX(OBJ_1.AVAILABLEFORSUPPLYDATE) AVAILABLEFORSUPPLYDATE",
        "RIGHT(OBJ_1.BATCH, 4) BATCH",
        "SUM(OBJ_1.QUANTITY) QUANTITY_SUM",
        "MAX(OBJ_1.STORE) STORE",
        "MAX(OBJ_2.STARTDATE) STARTDATE",
        "MAX(OBJ_2.TRANSACTIONCODE) TRANSACTIONCODE"
      ],
      "group_by" : [
        "OBJ_1.ITEMID",
        "OBJ_1.LOCATIONID",
        "RIGHT(OBJ_1.BATCH, 4)"
      ],
      "join" : [
        {
          "collection_id" : 1,
          "alias" : "OBJ_2",
          "key" : "ITEMID",
          "object" : "ZAMBONI_DB.ZAMBONI_SRC.INVENTORY_TRANSACTIONS"
        }
      ],
      "order_by_pos" : [1,2,3,7,6],
      "settings" : {
        "target_interval" : "minute",
        "target_lag" : 1440,
        "warehouse" : "xs_wh",
        "when_matched" : [],
        "when_not_matched" : []
      },
      "source" : {
        "collection_id" : 1,
        "alias" : "OBJ_1",
        "key" : "ITEMID",
        "object" : "ZAMBONI_DB.ZAMBONI_SRC.INVENTORY_ON_HANDS"
      },
      "target" : {
        "collection_id" : 2,
        "alias" : "TGT_1",
        "key" : null,
        "object" :
"ZAMBONI_DB.ZAMBONI_TGT.INVENTORY_BY_TRANSACTION_INCREMENTAL"
      },
      "where" : [
        {
          "attr_1" : "OBJ_1.LOCATIONID",
          "operator" : "=",
          "attr_2" : "'LDC1'",
          "condition" : "AND"
        },
        {
          "attr_1" : "OBJ_1.QUANTITY",
          "operator" : ">",
        "attr_2" : "500",
        "condition" : ""
      }
    ],
    "mapping" : [
      {
        "source_attr" : "RECORD_ID",
        "target_attr" : "RECORD_ID",
        "merge_on" : "Y",
        "update" : "N",
        "insert" : "Y"
      },
      {
        "source_attr" : "ITEMID",
        "target_attr" : "ITEM_ID",
        "merge_on" : "N",
        "update" : "Y",
        "insert" : "Y"
      },
      {
        "source_attr" : "LOCATIONID",
        "target_attr" : "LOCATION_ID",
        "merge_on" : "N",
        "update" : "Y",
        "insert" : "Y"
      },
      {
        "source_attr" : "PROJECT",
        "target_attr" : "PROJECT_NAME",
        "merge_on" : "N",
        "update" : "Y",
        "insert" : "Y"
        },
        {
          "source_attr" : "TYPE",
          "target_attr" : "TYPE",
          "merge_on" : "N",
          "update" : "Y",
          "insert" : "Y"
        },
        {
          "source_attr" : "AVAILABLEFORSUPPLYDATE",
          "target_attr" : "SUPPLY_DATE",
          "merge_on" : "N",
          "update" : "Y",
          "insert" : "Y"
        },
        {
          "source_attr" : "BATCH",
          "target_attr" : "BATCH_ID",
          "merge_on" : "N",
          "update" : "Y",
          "insert" : "Y"
        },
        {
          "source_attr" : "QUANTITY_SUM",
          "target_attr" : "QUANTITY_SUM",
          "merge_on" : "N",
          "update" : "Y",
          "insert" : "Y"
        },
        {
          "source_attr" : "STORE",
          "target_attr" : "STORE_NAME",
          "merge_on" : "N",
          "update" : "Y",
          "insert" : "Y"
        },
        {
          "source_attr" : "STARTDATE",
          "target_attr" : "START_DATE",
          "merge_on" : "N",
          "update" : "Y",
          "insert" : "Y"
        },
        {
          "source_attr" : "TRANSACTIONCODE",
          "target_attr" : "TRANSACTION_CODE",
          "merge_on" : "N",
          "update" : "Y",
          "insert" : "Y"
        }
      ],
      "labels" : ["label1", "label2"]
    }
  ]
}
$$)
);
```

Stored Procedure: MANAGE_DAG

Description:
This stored procedure adds/updates the DAG, to the PROCESS_DAG table, for a series of processes to be executed. This procedure should be called after the consumer defines the processes and order to be executed for a given DAG.

Parameters:

- **parent_process_name** (VARCHAR) - the name of the DAG.
- **child_processes** (VARIANT) - the processes to be executed, including order of execution.
- **labels** (ARRAY) - any descriptors that are applicable to the DAG.

Sample Call:

Unset

```
CALL MANAGE_DAG('dag_merge_inventory_by_transaction', PARSE_JSON('
{
  "child_processes" : [
    {
      "process_id" : 3,
      "process_name" : "target_incremental_merge_inventory_by_transaction_2",
      "process_order" : 2
    },
    {
      "process_id" : 2,
      "process_name" : "target_incremental_merge_inventory_by_transaction_1",
      "process_order" : 1
    }
  ]
}'), ARRAY_CONSTRUCT('label1','label2'));
```

Stored Procedure: CREATE_PROCESS

Description:
This stored procedure is called by the CREATE_PARENT_PROCESS stored procedure and executes the process passed to it. This procedure should be called once to create the process and either creates the initial object or performs the initial target update. In addition, it sets the refresh interval (when the target object is a dynamic table) or creates a task to update the target (when the target is a table).

Parameters:

- **parent_process_id** (NUMBER(38,0)) - the id of the DAG this process belongs to.
- **process_id** (NUMBER(38,0)) - the id of the process to be created.
- **run_id** (NUMBER(38,0)) - the id of the specific run of this process.
- **prev_process_name** (VARCHAR) - the name of the process that proceeds this process (if part of a multi-process DAG).
    - **NOTE**: this value is NULL/empty if this process is the first or only process in a DAG.

Sample Call:

Unset
```
CALL CREATE_PROCESS(1,2,1,'target_dt_inventory_by_transaction');
```

Stored Procedure: UPDATE_TARGET

Description:
This stored procedure is called by tasks created via the CREATE_PROCESS stored procedure to update the target table. This procedure executes the sql command passed to it, built from the process type's template in the PROCESS_TYPES table and the GET_SQL_JINJA function. The task calls this procedure at the refresh cadence defined in the process.

Parameters:

- **parent_process_id** (NUMBER(38,0)) - the id of the DAG this process belongs to.
- **process_id** (NUMBER(38,0)) - the id of the process to be executed.
- **sql_command** (NUMBER(38,0)) - the sql command that should be executed, when called.

Sample Call:

Unset
```
CALL UPDATE_TARGET(1,2,$$MERGE INTO
ZAMBONI_DB.ZAMBONI_TGT.INVENTORY_BY_TRANSACTION_INCREMENTAL t
        USING
            (
                SELECT

HASH(OBJ_1.ITEMID,OBJ_1.LOCATIONID,MAX(OBJ_1.PROJECT),MAX(OBJ_1.AVAILABLEFORSUP
PLYDATE),RIGHT(OBJ_1.BATCH,
4),SUM(OBJ_1.QUANTITY),MAX(OBJ_1.STORE),MAX(OBJ_2.STARTDATE),MAX(OBJ_2.TRANSACT
IONCODE)) RECORD_ID
                    ,OBJ_1.ITEMID
                    ,OBJ_1.LOCATIONID
                    ,MAX(OBJ_1.PROJECT) PROJECT
                    ,MAX(OBJ_1.TYPE) TYPE
                    ,MAX(OBJ_1.AVAILABLEFORSUPPLYDATE) AVAILABLEFORSUPPLYDATE
                    ,RIGHT(OBJ_1.BATCH, 4) BATCH
                    ,SUM(OBJ_1.QUANTITY) QUANTITY_SUM
                    ,MAX(OBJ_1.STORE) STORE
                    ,MAX(OBJ_2.STARTDATE) STARTDATE
                    ,MAX(OBJ_2.TRANSACTIONCODE) TRANSACTIONCODE
                FROM ZAMBONI_DB.ZAMBONI_SRC.INVENTORY_ON_HANDS OBJ_1
                JOIN ZAMBONI_DB.ZAMBONI_SRC.INVENTORY_TRANSACTIONS OBJ_2 ON
OBJ_1.ITEMID = OBJ_2.ITEMID
                WHERE
                    OBJ_1.LOCATIONID IN ('LDC2', 'LDC3') AND
                    OBJ_1.ITEMID <> 'CFGB09' AND
                    OBJ_1.AVAILABLEFORSUPPLYDATE <> '2023-10-30'
                GROUP BY
                    OBJ_1.ITEMID
                    ,OBJ_1.LOCATIONID
                    ,RIGHT(OBJ_1.BATCH, 4)
                ORDER BY
                    RECORD_ID
                    ,ITEMID
                    ,LOCATIONID
                    ,RIGHT(OBJ_1.BATCH, 4)
                    ,AVAILABLEFORSUPPLYDATE
            ) s ON
            t.RECORD_ID = s.RECORD_ID
        WHEN MATCHED
        THEN UPDATE SET
            t.ITEM_ID = s.ITEMID
            ,t.LOCATION_ID = s.LOCATIONID
            ,t.PROJECT_NAME = s.PROJECT
            ,t.TYPE = s.TYPE
            ,t.SUPPLY_DATE = s.AVAILABLEFORSUPPLYDATE
            ,t.BATCH_ID = s.BATCH
            ,t.QUANTITY_SUM = s.QUANTITY_SUM
            ,t.STORE_NAME = s.STORE
            ,t.START_DATE = s.STARTDATE
            ,t.TRANSACTION_CODE = s.TRANSACTIONCODE
        WHEN NOT MATCHED
        THEN INSERT
        (
            RECORD_ID
            ,ITEM_ID
            ,LOCATION_ID
            ,PROJECT_NAME
            ,TYPE
            ,SUPPLY_DATE
            ,BATCH_ID
            ,QUANTITY_SUM
```

```
            ,STORE_NAME
            ,START_DATE
            ,TRANSACTION_CODE
        )
        VALUES (
            s.RECORD_ID
            ,s.ITEMID
            ,s.LOCATIONID
            ,s.PROJECT
            ,s.TYPE
            ,s.AVAILABLEFORSUPPLYDATE
            ,s.BATCH
            ,s.QUANTITY_SUM
            ,s.STORE
            ,s.STARTDATE
            ,s.TRANSACTIONCODE
        )
        ;
$$);
```

Stored Procedure: CREATE_PARENT_PROCESS

Description:
This stored procedure calls the CREATE_PROCESS stored procedure for each process defined in the DAG, in the order specified. This procedure should be called when the consumer is ready to create the DAG.

Parameters:

- **parent_process_id** (NUMBER(38,0)) - the id of the DAG to create.

Sample Call:

```
Unset
CALL CREATE_PARENT_PROCESS(1);
```

Stored Procedure: UPDATE_PROCESS

Description:
This stored procedure is called by the UPDATE_PARENT_PROCESS stored procedure and either suspends or resumes the task that updates the process passed to it. This procedure should be called anytime a process needs to be suspended or resumed.
Parameters:
- **parent_process_id** (NUMBER(38,0)) - the id of the DAG this process belongs to.
- **process_id** (NUMBER(38,0)) - the id of the process to be updated.
- **run_id** (NUMBER(38,0)) - the id of the specific run of this process.
- **action** (VARCHAR) - SUSPEND or RESUME the task associated with the process.

Sample Call:

```
Unset
CALL UPDATE_PROCESS(1,2,1,'suspend');
```

Stored Procedure: UPDATE_PARENT_PROCESS

Description:
This stored procedure calls the UPDATE_PROCESS stored procedure for each process defined in the DAG, in the order specified. This procedure should be called when the consumer needs to suspend or resume each process' task .

Parameters:
- **parent_process_id** (NUMBER(38,0)) - the id of the DAG to create.
- **action** (VARCHAR) - SUSPEND or RESUME the task associated with the process.

Sample Call:

```
Unset
CALL UPDATE_PARENT_PROCESS(1, 'suspend');
```

Function: GET_SQL_JINJA

Description:
This function executes SQL commands based on the process type (jinja template) and process (parameters) passed to it. This function is called by the CREATE_PROCESS stored procedure.

Parameters:

- **template** (STRING) - the jinja template, converted to string.
- **parameters** (VARIANT) - JSON containing parameter names and values to substitute in the jinja template.

Sample Call:

Unset
```
--test target_incremental_merge_insert template
SELECT ZAMBONI_DB.ZAMBONI_UTIL.GET_SQL_JINJA(
    (SELECT TEMPLATE FROM ZAMBONI_DB.ZAMBONI_METADATA.PROCESS_TYPES WHERE
LOWER(PROCESS_TYPE) = 'target_dynamic_table'),
    PARSE_JSON($$
    {
      "process_name" : "target_dt_inventory_by_transaction",
      "process_type_id" : 1,
      "distinct": true,
      "top": null,
      "columns" : [
        "OBJ_1.ITEMID",
        "OBJ_1.LOCATIONID",
        "MAX(OBJ_1.PROJECT) PROJECT",
        "MAX(OBJ_1.TYPE) TYPE",
        "MAX(OBJ_1.AVAILABLEFORSUPPLYDATE) AVAILABLEFORSUPPLYDATE",
        "RIGHT(OBJ_1.BATCH, 4) BATCH",
        "SUM(OBJ_1.QUANTITY) QUANTITY_SUM",
        "MAX(OBJ_1.STORE) STORE",
        "MAX(OBJ_2.STARTDATE) STARTDATE",
        "MAX(OBJ_2.TRANSACTIONCODE) TRANSACTIONCODE"
      ],
      "group_by" : [
        "OBJ_1.ITEMID",
        "OBJ_1.LOCATIONID",
        "RIGHT(OBJ_1.BATCH, 4)"
      ],
      "join" : [
        {
          "collection_id" : 1,
          "alias" : "OBJ_2",
          "object" : "ZAMBONI_DB.ZAMBONI_SRC.INVENTORY_TRANSACTIONS",
          "keys" : [
            {
              "attr_1" : "OBJ_1.ITEMID",
              "operator" : "=",
              "attr_2" : "OBJ_2.ITEMID",
              "condition" : ""
            }
          ]
        }
      ],
      "order_by_cols" : [
        "ITEMID",
        "LOCATIONID",
        "BATCH",
        "AVAILABLEFORSUPPLYDATE"
      ],
      "settings" : {
        "downstream" : true,
        "target_interval" : "hours",
        "target_lag" : 24,
        "warehouse" : "xs_wh"
      },
      "source" : {
        "collection_id" : 1,
        "alias" : "OBJ_1",
        "key" : "ITEMID",
        "object" : "ZAMBONI_DB.ZAMBONI_SRC.INVENTORY_ON_HANDS"
      },
      "target" : {
        "collection_id" : 2,
        "alias" : "OBJ3",
        "key" : null,
        "object" : "ZAMBONI_DB.ZAMBONI_TGT.INVENTORY_BY_TRANSACTION_DYNAMIC"
      },
      "where" : [
        {
          "attr_1" : "OBJ_1.LOCATIONID",
          "operator" : "IN",
          "attr_2" : "('LDC2', 'LDC3')",
          "condition" : "AND"
        },
        {
          "attr_1" : "OBJ_1.ITEMID",
          "operator" : "!=",
          "attr_2" : "'CFGB09'",
          "condition" : "AND"
        },
        {
          "attr_1" : "OBJ_1.AVAILABLEFORSUPPLYDATE ",
          "operator" : "!=",
          "attr_2" : "'2023-10-30'",
          "condition" : ""
        }
      ],
      "mapping" : [],
      "labels" : ["label1", "label2"]
    }$$)
);
```

Stored Procedure: TEST_QUERY

Description:
This stored procedure executes any query passed to it, returning a table of up to the first 100 results. This procedure should be called when a consumer wants to test the output of their

process' source-to-target mappings. Ideally, the resulting table would be displayed in any UI built on top of Zamboni

Parameters:
- **sql_command** (VARCHAR) - the sql command to be executed.

Sample Call:

Unset
```
CALL TEST_QUERY($$SELECT
                OBJ_1.ITEMID
                ,OBJ_1.LOCATIONID
                ,MAX(OBJ_1.PROJECT) PROJECT
                ,MAX(OBJ_1.TYPE) TYPE
                ,MAX(OBJ_1.AVAILABLEFORSUPPLYDATE) AVAILABLEFORSUPPLYDATE
                ,RIGHT(OBJ_1.BATCH, 4) BATCH
                ,SUM(OBJ_1.QUANTITY) QUANTITY_SUM
                ,MAX(OBJ_1.STORE) STORE
                ,MAX(OBJ_2.STARTDATE) STARTDATE
                ,MAX(OBJ_2.TRANSACTIONCODE) TRANSACTIONCODE
            FROM ZAMBONI_DB.ZAMBONI_SRC.INVENTORY_ON_HANDS OBJ_1
            JOIN ZAMBONI_DB.ZAMBONI_SRC.INVENTORY_TRANSACTIONS OBJ_2 ON
OBJ_1.ITEMID = OBJ_2.ITEMID
            WHERE
                OBJ_1.LOCATIONID IN ('LDC2', 'LDC3') AND
                OBJ_1.ITEMID <> 'CFGB09' AND
                OBJ_1.AVAILABLEFORSUPPLYDATE <> '2023-10-30'
            GROUP BY
                OBJ_1.ITEMID
                ,OBJ_1.LOCATIONID
                ,RIGHT(OBJ_1.BATCH, 4)
            ORDER BY
                ITEMID
                ,LOCATIONID
                ,RIGHT(OBJ_1.BATCH, 4)
                ,AVAILABLEFORSUPPLYDATE
$$);
```

Appendix

Sample Collections JSON Payload

JavaScript

```
{
   "objects": [
     {
       "alias": "OBJ_1",
       "columns": [
         "ITEMID",
         "LOCATIONID",
         "PROJECT",
         "TYPE",
         "AVAILABLEFORSUPPLYDATE",
         "BATCH",
         "UOM",
         "QUANTITY",
         "PROCESSTYPE",
         "EXPIRATIONDATE",
         "SITEOWNER",
         "ITEMOWNER",
         "ONHANDPOSTDATETIME",
         "MEASURE",
         "NODETYPE",
         "LOB",
         "LOTNUMBER",
         "STORE",
         "CTOITEMID",
         "CTOBOMID"
       ],
       "object_id": "1"
     },
    {
      "alias": "OBJ_2",
      "columns": [
        "STARTDATE",
        "TRANSACTIONCODE"
      ],
      "object_id": 2
    }
  ]
}
```

Sample Create Process: Create Dynamic Table JSON Payload

JavaScript

```
{
  "process_name" : "target_dt_inventory_by_transaction",
  "process_type_id" : 1,
  "distinct": true,
  "top": null,
  "columns" : [
    "OBJ_1.ITEMID",
    "OBJ_1.LOCATIONID",
    "MAX(OBJ_1.PROJECT) PROJECT",
    "MAX(OBJ_1.TYPE) TYPE",
    "MAX(OBJ_1.AVAILABLEFORSUPPLYDATE) AVAILABLEFORSUPPLYDATE",
    "RIGHT(OBJ_1.BATCH, 4) BATCH",
    "SUM(OBJ_1.QUANTITY) QUANTITY_SUM",
    "MAX(OBJ_1.STORE) STORE",
    "MAX(OBJ_2.STARTDATE) STARTDATE",
    "MAX(OBJ_2.TRANSACTIONCODE) TRANSACTIONCODE"
  ],
  "group_by" : [
    "OBJ_1.ITEMID",
    "OBJ_1.LOCATIONID",
    "RIGHT(OBJ_1.BATCH, 4)"
  ],
  "join" : [
    {
      "collection_id" : 1,
      "alias" : "OBJ_2",
      "object" : "ZAMBONI_DB.ZAMBONI_SRC.INVENTORY_TRANSACTIONS",
      "keys" : [
        {
          "attr_1" : "OBJ_1.ITEMID",
          "operator" : "=",
          "attr_2" : "OBJ_2.ITEMID",
          "condition" : ""
        }
      ]
    }
  ],
  "order_by_cols" : [
    "ITEMID",
    "LOCATIONID",
    "BATCH",
    "AVAILABLEFORSUPPLYDATE"
  ],
  "settings" : {
    "downstream" : true,
    "target_interval" : "hours",
    "target_lag" : 24,
    "warehouse" : "xs_wh"
  },
  "source" : {
    "collection_id" : 1,
    "alias" : "OBJ_1",
    "key" : "ITEMID",
    "object" : "ZAMBONI_DB.ZAMBONI_SRC.INVENTORY_ON_HANDS"
  },
  "target" : {
    "collection_id" : 2,
    "alias" : "OBJ3",
    "key" : null,
    "object" : "ZAMBONI_DB.ZAMBONI_TGT.INVENTORY_BY_TRANSACTION_DYNAMIC"
  },
  "where" : [
    {
      "attr_1" : "OBJ_1.LOCATIONID",
      "operator" : "IN",
      "attr_2" : "('LDC2', 'LDC3')",
      "condition" : "AND"
    },
    {
      "attr_1" : "OBJ_1.ITEMID",
      "operator" : "!=",
      "attr_2" : "'CFGB09'",
      "condition" : "AND"
    },
    {
      "attr_1" : "OBJ_1.AVAILABLEFORSUPPLYDATE ",
      "operator" : "!=",
      "attr_2" : "'2023-10-30'",
      "condition" : ""
    }
  ],
    "mapping" : [],
    "labels" : ["label1", "label2"]
  }
```

Sample Create Process: Incremental Merge/Insert JSON Payload

JavaScript

```
{
  "process_name" : "target_incremental_merge_inventory_by_transaction_1",
  "process_type_id" : 2,
  "distinct": false,
  "top": 1000,
  "columns" : [

"HASH(OBJ_1.ITEMID,OBJ_1.LOCATIONID,MAX(OBJ_1.PROJECT),MAX(OBJ_1.AVAILABLEFORSU
PPLYDATE),RIGHT(OBJ_1.BATCH,
4),SUM(OBJ_1.QUANTITY),MAX(OBJ_1.STORE),MAX(OBJ_2.STARTDATE),MAX(OBJ_2.TRANSACT
IONCODE)) RECORD_ID" ,
    "OBJ_1.ITEMID",
    "OBJ_1.LOCATIONID",
    "MAX(OBJ_1.PROJECT) PROJECT",
    "MAX(OBJ_1.TYPE) TYPE",
    "MAX(OBJ_1.AVAILABLEFORSUPPLYDATE) AVAILABLEFORSUPPLYDATE",
    "RIGHT(OBJ_1.BATCH, 4) BATCH",
    "SUM(OBJ_1.QUANTITY) QUANTITY_SUM",
    "MAX(OBJ_1.STORE) STORE",
    "MAX(OBJ_2.STARTDATE) STARTDATE",
    "MAX(OBJ_2.TRANSACTIONCODE) TRANSACTIONCODE"
  ],
  "group_by" : [
    "OBJ_1.ITEMID",
    "OBJ_1.LOCATIONID",
    "RIGHT(OBJ_1.BATCH, 4)"
  ],
  "join" : [
    {
      "collection_id" : 1,
      "alias" : "OBJ_2",
      "key" : "ITEMID",
      "object" : "ZAMBONI_DB.ZAMBONI_SRC.INVENTORY_TRANSACTIONS"
    }
  ],
  "order_by_pos" : [1,2,3,7,6],
  "settings" : {
    "target_interval" : "minute",
    "target_lag" : 1440,
    "warehouse" : "xs_wh",
    "when_matched" : [],
    "when_not_matched" : []
  },
  "source" : {
    "collection_id" : 1,
    "alias" : "OBJ_1",
    "key" : "ITEMID",
    "object" : "ZAMBONI_DB.ZAMBONI_SRC.INVENTORY_ON_HANDS"
  },
  "target" : {
    "collection_id" : 2,
    "alias" : "TGT_1",
    "key" : null,
    "object" : "ZAMBONI_DB.ZAMBONI_TGT.INVENTORY_BY_TRANSACTION_INCREMENTAL"
  },
  "where" : [
    {
      "attr_1" : "OBJ_1.LOCATIONID",
      "operator" : "IN",
      "attr_2" : "('LDC2', 'LDC3')",
      "condition" : "AND"
    },
    {
      "attr_1" : "OBJ_1.ITEMID",
      "operator" : "!=",
      "attr_2" : "'CFGB09'",
      "condition" : "AND"
    },
    {
      "attr_1" : "OBJ_1.AVAILABLEFORSUPPLYDATE ",
      "operator" : "!=",
      "attr_2" : "'2023-10-30'",
      "condition" : ""
    }
  ],
  "mapping" : [
    {
      "source_attr" : "RECORD_ID",
      "target_attr" : "RECORD_ID",
      "merge_on" : "Y",
      "update" : "N",
      "insert" : "Y"
    },
    {
      "source_attr" : "ITEMID",
      "target_attr" : "ITEM_ID",
      "merge_on" : "N",
      "update" : "Y",
      "insert" : "Y"
    },
    {
      "source_attr" : "LOCATIONID",
      "target_attr" : "LOCATION_ID",
      "merge_on" : "N",
      "update" : "Y",
      "insert" : "Y"
    },
    {
      "source_attr" : "PROJECT",
      "target_attr" : "PROJECT_NAME",
      "merge_on" : "N",
      "update" : "Y",
      "insert" : "Y"
    },
    {
      "source_attr" : "TYPE",
      "target_attr" : "TYPE",
      "merge_on" : "N",
      "update" : "Y",
      "insert" : "Y"
    },
    {
      "source_attr" : "AVAILABLEFORSUPPLYDATE",
      "target_attr" : "SUPPLY_DATE",
      "merge_on" : "N",
      "update" : "Y",
      "insert" : "Y"
    },
    {
      "source_attr" : "BATCH",
    "target_attr" : "BATCH_ID",
    "merge_on" : "N",
    "update" : "Y",
    "insert" : "Y"
  },
  {
    "source_attr" : "QUANTITY_SUM",
    "target_attr" : "QUANTITY_SUM",
    "merge_on" : "N",
    "update" : "Y",
    "insert" : "Y"
  },
  {
    "source_attr" : "STORE",
    "target_attr" : "STORE_NAME",
    "merge_on" : "N",
    "update" : "Y",
    "insert" : "Y"
  },
  {
    "source_attr" : "STARTDATE",
    "target_attr" : "START_DATE",
    "merge_on" : "N",
    "update" : "Y",
    "insert" : "Y"
  },
  {
    "source_attr" : "TRANSACTIONCODE",
    "target_attr" : "TRANSACTION_CODE",
    "merge_on" : "N",
    "update" : "Y",
    "insert" : "Y"
    }
  ],
  "labels" : ["label1", "label2"]
}
```

Sample Manage Process DAG JSON Payload

JavaScript

```
{
 "child_processes" : [
   {
     "process_id" : 1,
     "process_name" : "target_dynamic_inventory_by_transaction",
     "process_order" : 1
   },
   {
     "process_id" : 2,
     "process_name" :
"target_incremental_insert_inventory_by_transaction_incremental",
     "process_order" : 2
   }
 ]
}
```