

Retek - Batch Module Design

Merchandising 10.0

November 2000

Functional Area

615 – Zone Change Effective Date

Module Affected

New – Price Transactions Extract for Store Zone Changes (pctranex.pc)

Design Overview

The Price Transactions Extract for Store Zone Changes module performs the final steps in processing pricing transactions that are created when a store is moved from one zone to another. This module will extract the price changes generated from store zone change that will become effective tomorrow and update the RMS tables to reflect the new prices. It will select all items that exist at the store that was moved and are associated with the zone group in which the move was made. Then, updates are made to the item/store prices, along with inserts into price history, supplier history, and transaction-level stock ledger tables.

If the Batch with Online Users indicator is 'Y', meaning users can be in the system at any given time, the program should determine if any of the records that will be processed are locked by the users. If there are locks encountered, the locked records will be written to a reject table, and these records will be reprocessed after the batch has run. The program will exit with the appropriate exit code, indicating that the program terminated successfully but with locking errors encountered. The rejected records will be checked for locks once again during the reprocessing. If there are no more locks encountered, these records will be processed and deleted from the reject table. If locks are still encountered, no new records will be written to the reject table. The locked records will be skipped, and will wait for reprocessing.

Scheduling Constraints*Pre/Post Logic Description*

Processing Cycle: Phase 3 - Daily

Scheduling Diagram: After the pctrandn.pc module is run

Pre-Processing: N/A

Post-Processing: Processed records will be deleted from the price_batch_tran table.

Threading Scheme: Store

TABLE	INDEX	SELECT	INSERT	UPDATE	DELETE
ITEM_SUPPLIER	X	X			
PERIOD		X			
PRICE_BATCH_TRAN		X			
PRICE_HIST			X		
PRICE_ZONE	X				X
ITEM_ZONE_PRICE	X	X			X
PRICE_ZONE_GROUP_STORE	X	X		X	
SUP_DATA			X		
SYSTEM_OPTIONS		X			
ITEM_MASTER	X	X			

Retek - Batch Module Design

Merchandising 10.0

November 2000

ITEM LOC	X	X		X	
PCTRANEX_LOCK_REJECT		X	X		X
BATCH_LOCK_LOG		X	X		X

Restart Recovery*Logical Unit of Work (recommended Commit check points)**Driving Cursor*

The restart string is new zone + store + old zone + item identifier. The restart is split between two cursors, with c_tran being the main cursor and c_item. The commit counter should be set to 10,000 records.

Driving Cursor:

```
/* Cursor to retrieve records from price_batch_tran table */
EXEC SQL DECLARE c_tran CURSOR FOR
    SELECT pbt.zone_group_id,
           pbt.old_zone_id,
           pbt.store,
           pbt.new_zone_id,
           pbt.new_primary_store
    FROM v_restart_store rv,
         price_batch_tran pbt
 WHERE pbt.effective_date = TO_DATE(:os_tomorrow, 'YYYYMMDD')
       AND rv.driver_value = pbt.store
       AND rv.driver_name = :ps_restart_driver_name
       AND rv.thread_val = :pi_restart_thread_val
       AND rv.num_threads = :pi_restart_num_threads
       AND (pbt.zone_group_id > NVL(:ps_restart_zone_group, -999)
            OR (pbt.zone_group_id = :ps_restart_zone_group
                AND (pbt.old_zone_id > :ps_restart_zone_id
                    OR (pbt.old_zone_id = :ps_restart_zone_id
                        AND (pbt.store >= :ps_restart_store))))))
 ORDER BY pbt.zone_group_id, pbt.old_zone_id, pbt.store;
```

Program Flow*Structure Chart*

N/A

Shared Modules*Listing of all externally referenced functions and Stored procedures and description of usage*

PRCCHGTYP – Calculates the tran_type for the tran_data insert.

STKLEDGR_SQL.TRAN_DATA_INSERT – Inserts a record into tran_data.

SUPP_ITEM_SQL.GET_PRI_SUP_COST – Gets the primary supplier cost for an item.

ITEMLOC_QTY_SQL.GET_TRANSFER – Determines stock amounts in transit (for tran_data insert).

Retek - Batch Module Design

Merchandising 10.0

November 2000

Function Level Description

All database interactions required and error handling considerations

item_processing()

This function is called from the process function. Information for the item is now fetched and processed in this function. If the item exists at the current store, processing continues. If it does not, skip to the next item. If the fetched item is a pack and it is not orderable, set its unit_cost to 0, otherwise call SUPP_ITEM_SQL.GET_PRI_SUP_COST to get cost info for the pack. The old and new retail information, uncluding the new selling unit_retail and uom information, is fetched from item_zone_price based on the new and old zone information selected from the price_batch_tran table. If the single unit retail price changes then the item_loc retail information is updated (including the new selling_unit_retail and uom fields) to hold the new values. If either the single or multi-unit price changes and the stock_qty is greater than zero (which will not occur for packs), then insert_tran() is called to insert records into tran_data, and create_price_hist is called to insert price_hist records. If neither retails change then the item should be skipped, since it is not changing in price at the store when the store moves zones.

calc_tran_type()

This function is called from the item_processing function and calls the stored procedure prcchgtype to determine the tran_type to be inserted into the tran_data table.

insert_tran()

First this function calls calc_tran_type() to determine the type of tran_data insert. The store's stock on hand and the quantity in transit must be added to determine the stock_qty, so the ITEMLOC_QTY_SQL.GET_TRANSFER stored procedure is called. If the stock_qty is positive the STKLEDGR_SQL.TRAN_DATA_INSERT package is called to insert the tran_data record. If the price change is a markup or markdown cancellation then supplier information is inserted into sup_data with a transaction type of 10 (markdown at retail). Because the transaction type is a markdown the markup amount should be multiplied by -1 to get the appropriate sign for the insert (to effectively create a markup). A similar process occurs for the second set of tran_types and tran_type amounts (for multi_unit information) to insert tran_data and sup_data records.

init()

In addition to performing standard restart tasks, the current date is selected from the period table, and the std_av_ind is selected from the system_options table to be used in selecting the correct unit_cost in the item_processing function for insert into the price_hist table in the create_price_hist function. The btch_w_usr_ind field will also be selected to determine the environment set up that is being used. If the Batch with Online Users indicator is 'Y', a cursor will be declared and used to check if there are existing records in the pctranex_lock_reject table. If the program is on a restart, another cursor will check if there are records existing in the batch_lock_log table. If there is an existing record in the table, a flag will be set to ensure that the appropriate exit code will be used. Otherwise, the record (if any), in the batch_lock_log table will be deleted. If the pi_thread_exists flag equals zero at anytime, this means that for the thread that is currently being processed, no reject records exist and in this case, no processing should occur. Instead, the program should exit successfully.

process()

Price_batch_tran is the driving table for the program. Store Zone Changes are selected in this function based on the effective date of the zone change. If today's date (vdate) is the day before the effective date, then the store information including new and old zone information will be selected in this function. If the zone change effective date is tomorrow then call the update_new_zone function to actually move the store to the new zone and the delete_old_zone function is called, which will delete the old zone information. Next, call the item_processing function for each zone change that will be effective tomorrow. Each call will be made passing in the item type parameter. The item_processing function will execute one of three cursors (depending on the item type) to retrieve all of the items in the zone group of the zone change.

Retek - Batch Module Design

Merchandising 10.0

November 2000

Prior to calling these functions, checks are done to determine if the Batch with Online Users indicator is 'Y'. If it is set up, there are additional conditions to see if the run is a restart and if there are existing records in the pctranex_lock_reject table. If there are existing records, and the run is not a restart, the driving cursor with joins to the reject table will be used to fetch data from the driving table for reprocessing. The tables price_zone_group_store and item_loc will also be checked for record locks. Locked records encountered are written to the batch_lock_log table. The records fetched from the driving cursor which are related to the locked records are written to the reject table, and will be reprocessed after the program completes.

create_price_hist()

This function is called from the item_processing function and inserts a record into price_hist. A different transaction type will be inserted depending on the type of price that changed: 4 for single unit price changes only, 10 for multi-unit price changes only, and 11 for single and multi-unit price changes.

delete_old_zone()

This function is called from the process function and will delete rows from zone tables where stores no longer exist in a zone_id. First the price_zone_group_store table is queried for all the stores in the old zone_id. If there are no records then rows with the old zone id/zone group are deleted from the appropriate zone tables: item_zone_price and price_zone.

update_new_zone()

This function is also called from the process function and will update the price_zone_group_store table for each store changing zones, setting the zone id equal to the new zone id on the price_batch_tran table.

final ()

This function will perform the standard restart/recovery close logic.

check_lock_row()

This function will check for locked records on the price_zone_group_store and item_loc tables. If locked records are encountered, a flag will be set, indicating that locked records were encountered, and another flag is set to determine which table to select from when inserting into the batch_lock_log table.

I/O Specification

All files layouts input and output

None.

Technical Issues

None.

Test Scenarios

1. Run program and ensure that only price_batch_tran records with an effective date that is equal to tomorrow are processed.
2. Only ITEM's that meet the following criteria should have been updated:
 - Must belong in the zone group of the zone change.
 - Are stocked at the store changing zones.
 - Item/Store is not on clearance.
 - Have a different unit retail in the old zone than in the new zone.
3. Price_hist records are inserted for the price changes for approved items generated as a result of the store zone change.
4. Tran_data records are inserted for each approved item changing price at the store changing zones.

Retek - Batch Module Design

Merchandising 10.0

November 2000

5. Sup_data records are inserted for each approved items at the dept/supplier affected by the price_changes.
6. Price_zone_group_store records are updated for stores changing zones, setting the zone id equal to the new zone id from the price_batch_tran table.
7. The driving cursor is used to fetch data if: (1) the run is not a restart and there are no records in the reject table, (2) the run is a restart and there are records in the reject table, and (3) the run is a restart and there are no records in the reject table.
8. The reject driving cursor is used to fetch data if the run is not a restart and there are no records in the reject table.