

Retek Clearance Pricing Update [pccext]

Design Overview

The function of this program is to extract the new clearance event retail pricing from the clearance suspense tables and update the retail information in Retek. Clearance events that are to take place on the following business day are selected and the appropriate tables are updated with the new pricing information and with a new clearance indicator value. Transaction-level stock ledger, price history, supplier history and clearance sell through records are also written. If an item/location is on replenishment, using the RMS replenishment functionality, then when the item/location goes on clearance, it is taken off of replenishment.

If the Batch with Online Users indicator is set to 'Y', it will bulk lock the item_loc records first to check for record locking for a given set of records that were retrieved using the clearance number. If there were no lock encountered, it will bulk lock the repl_day and repl_item_loc records to check for record locking for a given set of records that were retrieved using the clearance number. The record lock checking for the repl_day and repl_item_loc tables are done only when the system_options.contract_ind is 'N'. The record lock checking for the three tables is done before performing the necessary updates. If any of the records in the item_loc, repl_day or repl_item_loc tables are locked by another application, they will not be processed, thus an update to the item_loc table and a delete to the repl_day and repl_item_loc tables given that the system_options.contract_ind is 'N' for the set of records will not happen. Otherwise, the records will be processed and an update to the item_loc table and a delete to the repl_day and repl_item_loc tables for the set of records will happen.

| Table | Index | Select | Insert | Update | Delete |
|-----------------------------|--------------|---------------|---------------|---------------|---------------|
| PERIOD | No | Yes | No | No | No |
| CLEAR_SUSP_DETAIL | No | Yes | No | No | No |
| CLEAR_SUSP_HEAD | No | Yes | No | Yes | No |
| ITEM_LOC | No | Yes | No | Yes | No |
| ITEM_LOC_SOH | No | Yes | No | No | No |
| ITEM_MASTER | No | Yes | No | No | No |
| ITEM_SUPPLIER | No | Yes | No | No | No |
| CLEAR_SELL_THRGH | No | No | Yes | No | No |
| TRAN_DATA | No | No | Yes | No | No |
| SUP_DATA | No | No | Yes | No | No |
| PRICE_HIST | No | No | Yes | No | No |
| PRICE_ZONE | No | No | Yes | No | No |
| PRICE_ZONE_GROUP_STORE | Yes | Yes | No | No | No |
| REPL_DAY | No | No | No | No | Yes |
| REPL_ITEM_LOC | No | No | No | No | Yes |
| V_RESTART_CLEARANCE | No | Yes | No | No | No |
| SYSTEM_OPTIONS | No | Yes | No | No | No |
| BATCH_LOCK_LOG | No | No | Yes | No | Yes |
| CLEARANCE_EXTRACT_LOCK_TEMP | No | Yes | Yes | No | Yes |
| PCCEXT_LOCK_REJECT | No | Yes | Yes | No | Yes |

Indexes: CLEAR_SUSP_HEAD(reason)

ITEM_MASTER(standard_uom)

ITEM_SUPPLIER(supplier)

PRICE_ZONE_GROUP_STORE(zone_group_id, zone_id),
(store)

Scheduling Constraints

| | |
|---------------------|-----------------------------------------------------------|
| Processing Cycle: | Phase 3 |
| Scheduling Diagram: | N/A |
| Pre-Processing: | pctranex should complete before the pccext module begins. |
| Post-Processing: | pccrxt should not start until pccext has completed. |
| Threading Scheme: | Clearance |

Restart Recovery

```
/* Cursor to process clearance events (clear_susp_head) */
EXEC SQL DECLARE c_clear CURSOR FOR
SELECT csh.clearance,
       csh.reason,
       csh.status,
       csd.item,
       csd.unit_retail,
       csd.selling_uom,
       im.standard_uom,
       TO_CHAR(csd.active_date, 'DDMMYYYY'),
       pzgs.store,
       pz.currency_code,
       im.pack_ind,
       NVL(ilo.unit_retail,0),
       ils.stock_on_hand,
       NVL(ils.unit_cost, 0),
       ROWIDTOCHAR(ilo.rowid),
       ROWIDTOCHAR(csh.rowid),
       '||TO_CHAR(csh.clearance)||'
       '||csd.item||'
       '||TO_CHAR(pzgs.store)
FROM v_restart_clearance rv,
     price_zone_group_store pzgs,
     price_zone pz,
     clear_susp_detail csd,
     clear_susp_head csh,
     item_master im,
     item_loc ilo,
     item_loc_soh ils
WHERE csh.status      = 'A'
AND csh.clearance     = csd.clearance
AND csd.active_date   = to_date(:ps_tomorrow, 'DDMMYYYY')
AND csd.zone_group_id = pzgs.zone_group_id
AND csd.zone_id       = pzgs.zone_id
AND csd.zone_group_id = pz.zone_group_id
```

```
AND csd.zone_id    = pz.zone_id
AND csd.item       = im.item
AND im.item_level  = im.tran_level
AND im.status      = 'A'
AND im.pack_ind    = 'N'
    AND ilo.item    = csd.item
AND ilo.loc        = pzgs.store
AND ilo.status in ('A', 'C')
AND ilo.item       = ils.item
AND ilo.loc        = ils.loc
AND rv.driver_value = csh.clearance
AND rv.num_threads = TO_NUMBER(:ps_restart_num_threads)
AND rv.thread_val   = TO_NUMBER(:ps_restart_thread_val)
AND (csh.clearance > NVL(:ps_restart_clearance, -999) OR
    (csh.clearance = :ps_restart_clearance AND
    (csd.item > :ps_restart_item OR
    (csd.item = :ps_restart_item AND
    (pzgs.store > :ps_restart_store))))))
ORDER BY csh.clearance,
        csd.item,
        pzgs.store;
```

Program Flow

N/A

Shared Modules

STKLEDGR_SQL.TRAN_DATA_INSERT

PRCCHGTYP

GET_STANDARD_UOM

UOM_SQL.CONVERT

CURRENCY_SQL.CONVERT

Function Level Description

Init()

Initialize restart recovery.
Determine vdate, contract_ind and currency_code.
Call function size_arrays to allocate memory size to arrays.

Process()

Select all clearance details with an active date of tomorrow. It will call the function get_item_info to obtain the clearance items dept, class, subclass and supplier. Function update_item_loc is then called to convert the new selling unit retail to the standard unit retail, compute the tran type, insert into tran_data, clear_sell_through, price_hist and sup_data, and update item_loc.

Get Item Info()

This function selects the items dept, class, subclass, and supplier and fetches them into variables to be inserted into tables later in program.

Update_Item_Loc()

This function selects unit retail, unit cost, and stock on hand info from the ITEM_LOC and ITEM_LOC_SOH tables. It then converts the new unit retail for clearance items to standard unit retail by calling the convert_new_unit_retail function. The function also calls functions calc_tran_type to compute tran type, insert_tran to insert into the TRAN_DATE and SUP_DATA tables, clear_sell_through to insert into the CLEAR_SELL_THROUGH table, and price_hist to insert into the PRICE_HIST table. The function also updates the ITEM_LOC table with the new selling uom, selling unit retail and standard unit retail.

Convert_New_Unit_Retail()

Converts the new selling unit retail to the standard unit retail by calling the UOM_SQL.CONVERT package.

Calc_Tran_Type()

This function computes the tran type by calling the stored procedure PRCCHGTYP.

Insert_Tran()

If the selected item's stock qty > 0, this function calls the package STKLEDGR_SQL.TRAN_DATA_INSERT to insert into the TRAN_DATA table. If the local currency is not the same as the primary currency, this function calls function convert_currency to convert to the primary currency. Then insert into the SUP_DATA table. If there is a second tran type and tran type amt, then the function calls STKLEDGR_SQL.TRAN_DATA_INSERT again, and inserts into the SUP_DATA table again.

Convert_Currency()

Function is called to compute the dollar amount in primary currency by calling the package CURRENCY_SQL.CONVERT.

Price_Hist()

This function inserts into the PRICE_HIST table.

Clear_Sell_Through()

This function inserts into the CLEAR_SELL_THROUGH table.

Post_Change()

This function deletes an item from the REPL_DAY and REPL_ITEM_LOC tables when the item goes on clearance.

Size_Arrays()

This function allocates memory sizes to the arrays that have been fetched into by the driving cursor.

`check_lock()`

This function will check if a given set of records in item_loc table are locked by another application. Furthermore, if the system_options.contract_ind is 'N', this function will also check if a given set of records in the repl_day and repl_item_loc tables are locked by another application. If at least one from the set of records is locked regardless of which table, this function will write a record locking error in the error log and it will return 54. Otherwise, it will return 0 and continue processing. This function is called for every set of records fetched, depending on the commit max counter.

Final()

This function closes restart recovery.

I/O Specification

N/A

Technical Issues

N/A