Oracle® Retail Merchandising System Operations Guide Addendum Release 11.0.7 March 2006

ORACLE

Contents

Preface

Oracle Retail Operations Guides are designed so that you can view and understand the application's 'behind-the-scenes' processing.

Audience

Anyone with an interest in developing a deeper understanding of the underlying processes and architecture supporting Oracle Retail Merchandising System (RMS) functionality will find valuable information in this guide. There are three audiences in general for whom this guide is written:

- Business analysts looking for information about processes and interfaces to validate the support for business scenarios within and other systems across the enterprise.
- System analysts and system operations personnel:
  - Who are looking for information about RMS's processes internally or in relation to the systems across the enterprise.
  - Who operate RMS regularly.
- Integrators and implementation staff with overall responsibility for implementing RMS.

Related Documents

You can find more information about this product in these resources:

- Oracle Retail Merchandising System Installation Guide
- Oracle Retail Merchandising System Release Notes
- Oracle Retail Merchandising System Data Model

Customer Support

- https://metalink.oracle.com

When contacting Customer Support, please provide:

- Product version and program/module name.
- Functional and technical description of the problem (include business impact).
- Detailed step-by-step instructions to recreate.
- Exact error message received.
- Screen shots of each step you take.

1

Stock Count Stock on Hand Updates [stkvar]

Design Overview

The stkvar.pc program updates the stock on hand and computes the total cost and total retail in the stake_prod_loc.

Tables Affected:

TABLE	INDEX	SELECT	INSERT	UPDATE	DELETE
CLASS	No	Yes	No	No	No
ITEM_LOC	Yes	No	No	Yes	No
ITEM_LOC_SOH	Yes	No	No	Yes	No
ITEM_MASTER	No	Yes	No	No	No
ITEM_SUPP_COUNTRY	No	Yes	No	No	No
ITEM_XFORM_DETAIL	No	Yes	No	No	No
ITEM_XFORM_HEAD	No	Yes	No	No	No
NWP	No	No	Yes	Yes	No
NWP_FREEZE_DATE	No	Yes	No	No	No
STAKE_CONT	No	Yes	No	No	Yes
STAKE_HEAD	No	Yes	No	No	No
STAKE_PROD_LOC	Yes	No	No	Yes	No
STAKE_QTY	No	Yes	No	No	No
STAKE_SKU_LOC	No	Yes	No	Yes	No
WH	No	Yes	No	No	No

Indexes:

STAKE_PROD_LOC (dept, store, wh, data_type)

This program updates the stock on hand for all items as a result of a stock take.

The program is driven by STAKE_CONT, in conjunction with STAKE_SKU_LOC where the ITEM, loc and cycle count on STAKE_SKU_LOC match those on STAKE_CONT, and where STAKE_CONT run_type = 'A' (for adjustment).

For each row retrieved from the above tables, the unit systems are processed as follows:

- An ITEM_LOC_SOH record is updated for every ITEM/loc combination. The new stock on hand = item_loc_soh.stock_on_hand - snapshot_stock_on_hand_qty (from the STAKE_SKU_LOC table) + the physical count quantity on STAKE_SKU_LOC. In addition, the pack_comp_soh field is updated on ITEM_LOC when a pack is processed for each component ITEM in the pack.
- Total cost and total retail are computed as the snapshot unit retail times the sum of the physical count quantity plus the snapshot in-transit (from the STAKE_SKU_LOC table).
- STAKE_PROD_LOC total cost and total retail amounts are updated with the total cost and total retail for each department, class, subclass, location combination that exists on the cycle count. A record for each is added. If a record already exists on the table, the total cost or retail amount value is adjusted to be the existing total cost or retail amount + the cycle count cost or retail. If no record exists, a new one is added to the table with the value of total cycle count cost or retail for the total cost or retail amount. If the stock ledger is designated not to include VAT on the SYSTEM_OPTIONS table, the total retail amount will have any VAT amount stripped from it.

Re-run:

- If this program terminates normally, ITEM_LOC, STAKE_QTY,
- ITEM_LOC_SOH, STAKE_PROD_LOC and STAKE_CONT must be recovered prior to restart. If this program terminates abnormally, restart without recovery.

The syntax for invoking this program is:

stkvar `userid/pswd [report_name ]`.

Here are some examples:

```
stkvar userid/pswd ( it will not produce any report.)
stkvar userid/pswd any.rpt ( it will produce a report, any.rpt. )
```

Input Specifications

```
EXEC SQL DECLARE c_item CURSOR FOR
SELECT /* ORDERED USE_HASH(stake_cont) FULL(stake_head)*/
       sc.item,
       sc.loc_type,
       sc.location,
       c.class_vat_ind,
       ROWIDTOCHAR(sc.ROWID),
       ssl.snapshot_on_hand_qty,
       NVL(ssl.snapshot_in_transit_qty,0),
       NVL(ssl.snapshot_unit_cost, 0),
       NVL(ssl.snapshot_unit_retail, 0),
       NVL(ssl.physical_count_qty, 0),
       ssl.pack_comp_qty,
       NVL(ssl.dept, 0),
       NVL(ssl.class, 0),
       NVL(ssl.subclass, 0),
       ROWIDTOCHAR(ssl.ROWID),
       im.pack_ind,
       TO_CHAR(sh.stocktake_date,'YYYYMMDD'),
       sh.stocktake_type,
       sh.cycle_count,
       ssl.xform_item_type,
       im.deposit_item_type,
       ';' || sc.item   ||
```

```
       ';' || TO_CHAR(sc.loc_type) ||
       ';' || TO_CHAR(sc.location)
  FROM stake_sku_loc ssl,
       stake_cont sc,
       stake_head sh,
       item_master im,
       class c,
       wh w,
       v_restart_all_locations rv
 WHERE sc.run_type                    = 'A'
   AND sc.item                        = ssl.item
   AND sc.loc_type                    = ssl.loc_type
   AND sc.location                    = ssl.location
   AND sc.cycle_count                 = ssl.cycle_count
   AND sh.cycle_count                 = ssl.cycle_count
   AND NVL(ssl.xform_item_type, ' ') <>'S'/*exclude sellable only items*/
   AND im.item                        = sc.item
   AND im.item_level                  = im.tran_level
   AND im.dept                        = c.dept
   AND im.class                       = c.class
   AND ssl.location                   = w.wh (+)
   AND (im.pack_ind     = 'N' OR
       (ssl.loc_type    = 'W' AND
        im.pack_ind     = 'Y' AND
        w.finisher_ind  = 'N'))
   AND rv.driver_value  = sc.location
   AND rv.driver_name   = :ora_restart_driver_name
   AND rv.num_threads   = TO_NUMBER(:ora_restart_num_threads)
   AND rv.thread_val    = TO_NUMBER(:ora_restart_thread_val)
   AND (sc.item           > NVL(:ora_restart_item,-999) OR
          (sc.item        = :ora_restart_item AND
             (sc.loc_type > TO_CHAR(:ora_restart_loc_type) OR
             (sc.loc_type = TO_CHAR(:ora_restart_loc_type) AND
             (sc.location  > TO_NUMBER(:ora_restart_location)))
             )
          )
       )
 ORDER BY sc.location,
          sc.loc_type,
          sc.item;
```

Scheduling Constraints

Processing Cycle:	PHASE 3
Scheduling Diagram:	N/A
Pre-Processing:	N/A
Post-Processing:	N/A
Threading Scheme:	LOCATION

Restart Recovery

This program will be threaded by department and will utilize restart/recovery logic based on item/loc_type/location.

Shared Modules

PRICING_ATTRIB_SQL.GET_RETAIL

2

Stock Count Explode [stkxpld]

Design Overview

Stock Count Explode to Item/Location for Stores/Warehouses/External Finishers. This program combined stkxplst and stkxplwh to create a new program [stkxpld]. The Stock Count Explode batch program sets up the stock count snapshot by populating the stocktake tables with department-class-subclass relationship for all items to be counted for a chosen location.

Tables Affected:

TABLE	INDEX	SELECT	INSERT	UPDATE	DELETE
ITEM_LOC	No	Yes	No	No	No
ITEM_MASTER	No	Yes	No	No	No
ITEM_XFORM_DETAIL	No	Yes	No	No	No
ITEM_XFORM_HEAD	No	Yes	No	No	No
PERIOD	No	Yes	No	No	No
STAKE_HEAD	Yes	Yes	No	No	No
STAKE_LOCATION	Yes	Yes	No	No	No
STAKE_PROD_LOC	Yes	No	Yes	No	No
STAKE_PRODUCT	Yes	Yes	No	No	No
STAKE_SKU_LOC	Yes	No	Yes	No	No
SUBCLASS	No	Yes	No	No	No
SYSTEM_OPTIONS	No	Yes	No	No	No

This program inserts into stake_sku_loc and stake_prod_loc tables for all valid item/loc and department/loc combinations for unit and dollar type stock counts, based on location, department, class and subclass information stored on stake_location and stake_product. The stock count can be requested at department, class or subclass level and exploded to Item/loc from any of these levels. Inserts are made into stake_prod_loc now at subclass/loc level. This program also inserts into stake_sku_loc and deletes stake_product for all unit only counts for warehouses. Orderable only transformed item, an indicator ‘O’ is inserted in xform_item_type column of stake_sku_loc table. Also, for sellable only transformed item, an indicator ‘S’ is used.

Program Flow

N/A

Shared Modules

N/A

Function Level Description

main()

- Standard Retek main function
- Validates input parameters
- Calls init, process and final
- Logs appropriate message

init()

- Initialize restart variables
- Collect system date information from PERIOD and SYSTEM_OPTIONS

process()

- The cycle count data with stocktake date equal to the threshold date, which is equal to the vdate plus the stake lockout days, should exist both in STAKE_HEAD and STAKE_LOCATION tables
- Open the product cursor
- process_product function callout for each data to be processed
- Fetch the product cursor into global variables
- if NO_DATA_FOUND in the cursor, break the loop

process_product()

- Insert processed data into STAKE_SKU_LOC
- For unit and dollar type of stock count (‘B’), insert data into STAKE_PROD_LOC

final()

- Restart/recovery close, close files

Input Specification

```
EXEC SQL DECLARE C_product CURSOR FOR
   SELECT h.cycle_count,
          h.loc_type,
          h.stocktake_type,
          l.location
     FROM stake_location l,
          stake_head h,
          v_restart_all_locations rv
    WHERE h.cycle_count = l.cycle_count
      AND h.stocktake_date = (TO_DATE(:threshold_date,'DDMMYYYY'))
      AND h.delete_ind = 'N'
      AND rv.driver_value = l.location
      AND rv.num_threads = TO_NUMBER(:ps_num_threads)
      AND rv.thread_val  = TO_NUMBER(:ps_thread_val)
      AND (h.cycle_count > NVL(TO_NUMBER(:ps_restart_cycle_count), -999) OR
             (h.cycle_count = TO_NUMBER(:ps_restart_cycle_count) AND
                 (l.location > TO_NUMBER(:ps_restart_location) )))
 ORDER BY h.cycle_count,
          l.location;
```

Scheduling Constraints

Processing Cycle:	PHASE 3
Scheduling Diagram:	N/A
Pre-Processing:	N/A
Post-Processing:	STKUPD.PC
Threading Scheme:	LOCATION

Restart Recovery

This program will be threaded by location and will utilize restart/recovery logic based on cycle_count/location

Technical Issues

N/A