

Retek RMS - Batch Module Design

Functional Area:	614 – Location Lists
Module:	Location List Batch Rebuild (lclrbld.pc)
Difficulty:	Easy New Batch

Design Overview

Location List Batch Rebuild (lclrbld.pc)

The Location List Rebuild program performs the rebuilding of all location lists that have static_ind = 'N' and batch_rebuild_ind = 'Y'. It calls package function LOCLIST_BUILD_SQL.REBUILD_LIST to generate LOC_LIST_DETAIL records based on the store and warehouse criteria on the LOC_LIST_CRITERIA table.

This program needs to be modified to work in an environment where the users can be in the system at any given time. The system_option table will be modified to introduce a new indicator named as Batch with Online Users (btch_w_usr_ind). If this indicator is set to 'Y' then system allows users to be online at anytime. This batch should be modified to check whether the the Batch with Online Users indicator is set to 'Y' before processing any rebuilds on loc_list_detail

The records that are skipped due to locking errors should be inserted into the reject table for processing in the next run of the batch cycle. This program needs to distinguish which records have been processed. In order to this, a new driving cursor for table loc_list_head and the reject table will be created. Once records exists in the new driving cursor, it will process the records from the new driving cursor otherwise, it will process records from the c_loc_list cursor.

Scheduling Constraints

Pre/Post Logic Description

Processing Cycle: Phase II or III. (This is kind of a standalone batch program right now. No other batch program is using the result of location list rebuild at the moment.)

Scheduling Diagram:

Pre-Processing:

Post-Processing:

Threading Scheme: LOC_LIST

Restart Recovery

Logical Unit of Work (recommended Commit check points)

Driving Cursor

The Logical Unit of Work for this program is location list number. Since this unit of work is unique, there is no need to worry about things like reallocating memory or staggered posting.

The following driving cursor is used (with restart/recovery):

```
select l.loc_list
from loc_list_head l,
     v_restart_loc_list rv,
where rv.driver_value = l.loc_list
```

Retek RMS - Batch Module Design

```
and l.static_ind = 'N'
and l.batch_rebuild_ind = 'Y'
and rv.driver_name = :os_restart_driver_name
and rv.thread_val = :oi_restart_thread_val
and rv.num_threads = :oi_restart_num_threads
and l.loc_list > NVL(:oi_restart_loc_list, -999)
order by l.loc_list;
```

Program Flow

Structure Chart

Shared Modules

Listing of all externally referenced functions and Stored procedures and description of usage

LOCLIST_BUILD_SQL.REBUILD_LIST – for rebuilding the LOC_LIST_DETAIL records for a given location list number.

Data Structures

Constructs used to hold and manage data within the program

Function Level Description

All database interactions required and error handling considerations

main() – Follows Retek standards for the main() function. Logs on to the database, calls init(), process(), and final().

int() – Calls restart_init() to start up restart recovery and bring back the bookmark string in case of a restart. Gets btch_w_usr_ind from system_options table. Deletes record from the batch_lock_log table. It also checks if there are previous locks in case of restart recovery.

process() – The controller for most of the processing in the program. This function opens and fetches from the driving cursor or the reject driving cursor depending on the reject table and btch_w_usr_ind. For each record fetched from the driving cursor/reject driving cursor, records will be inserted to a temporary table and will be checked for records locks in the check_lock function. If there is a lock, the LUW will be skipped and continue processing the next LUW otherwise, it will just call the rebuild_location_list() to rebuild location list for the given location list number. When the fetched array has been fully processed, restart_commit() should be called to commit the work, and update the restart/recovery variables.

check_lock() – Check locks by joining the temporary table and target table.

rebuild_loc_list() – Calls the package function LOCLIST_BUILD_SQL.REBUILD_LIST to rebuild LOC_LIST_DETAIL records for a given location list number.

final() – Calls restart_close() to wrap up restart recovery.

I/O Specification

All files layouts input and output

Retek RMS - Batch Module Design

Testing Scenarios

1. Create a few location list, some with static_ind = 'N' and batch_rebuild_ind = 'Y', some with static_ind = 'Y' and/or batch_rebuild_ind = 'N'.
2. Add store and/or warehouse criteria without generating the location list.
3. Run the batch program lclrbld.pc to generate location lists for all location lists with static_ind = 'N' and batch_rebuild_ind = 'Y'.
4. Add some stores and warehouses that will fit the criteria, run lclrbld.pc again. New stores and warehouses that fit the criteria should be added on the location list.
5. Remove some stores and warehouses that fit the criteria, run lclrbld.pc again. A new location list should be generated without the removed stores and warehouses.
6. Test array processing.
7. Test restart recovery.

Technical Issues

Need to add a global temporary table LOC_LIST_HEAD_LOCK_TEMP:

```
SELECT CHARTOROWID(lh.rowid)
  FROM loc_list_head lh,
       loc_list_head_lock_temp l
 WHERE lh.rowid = l.row_id
    FOR update nowait;
```

```
SELECT CHARTOROWID(ld.rowid)
  FROM loc_list_detail ld,
       loc_list_head_lock_temp l
 WHERE ld.loc_list = l.loc_list
    FOR update nowait;
```

```
SELECT CHARTOROWID(se.rowid)
  FROM sit_explode se,
       loc_list_head_lock_temp l
 WHERE se.loc_list = l.loc_list
    FOR update nowait;
```

```
SELECT CHARTOROWID(sc.rowid)
  FROM sit_conflict sc
 WHERE sc.location in (SELECT location
                      FROM sit_explode se,
                           loc_list_head_lock_temp l
                      WHERE ((se.itemloc_link_id = sc.itemloc_link_id)
                          OR (se.itemloc_link_id = sc.conflict_itemloc_link_id))
                          AND se.loc_list = l.loc_list)
    FOR update nowait;
```

Need to add a table BATCH_LOCK_LOG:

```
SELECT 1
  FROM batch_lock_log
 WHERE program_name = :g_s_restart_name
```

Retek RMS - Batch Module Design

```
AND thread_val = TO_NUMBER(:ps_restart_thread_val)
AND rownum = 1;
```

Need to create a view V_RESTART_LOC_LIST:

```
select distinct rc.driver_name driver_name,
               rc.num_threads num_threads,
               l.loc_list driver_value,
               restart_thread_return(l.loc_list, rc.num_threads) thread_val
from restart_control rc,
     loc_list_head l
where rc.driver_name = 'LOC_LIST';
```

Need to add a record to RESTART_CONTROL for program lclrbld:

```
program_name: lclrbld
program_desc: location list batch rebuild
driver_name: LOC_LIST
num_threads: 1
update_allowed: Y
process_flag: T
commit_max_ctr: 100
```

Need to add records to RESTART_PROGRAM_STATUS for program lclrbld. The number of records added should be equal to the num_threads in RESTART_CONTROL table for program lclrbld:

```
restart_name: lclrbld
thread_val: 1
start_time: Null
program_name: lclrbld
program_status: ready for start
restart_flag: Null
finish_time: Null
current_pid: Null
current_operator_id: Null
err_message: Null
```

When testing, use num_threads of at least 2.

Please document the table changes in changes.xls.