

Recommended Order Quantity [ociroq.c]

Design Overview

The purpose of this batch program is to call the PL/SQL packages used to calculate the Net Inventory position of the items on replenishment. The results are stored in the database to be used by REQEXT (Item Requisition Extraction).

Scheduling Constraints

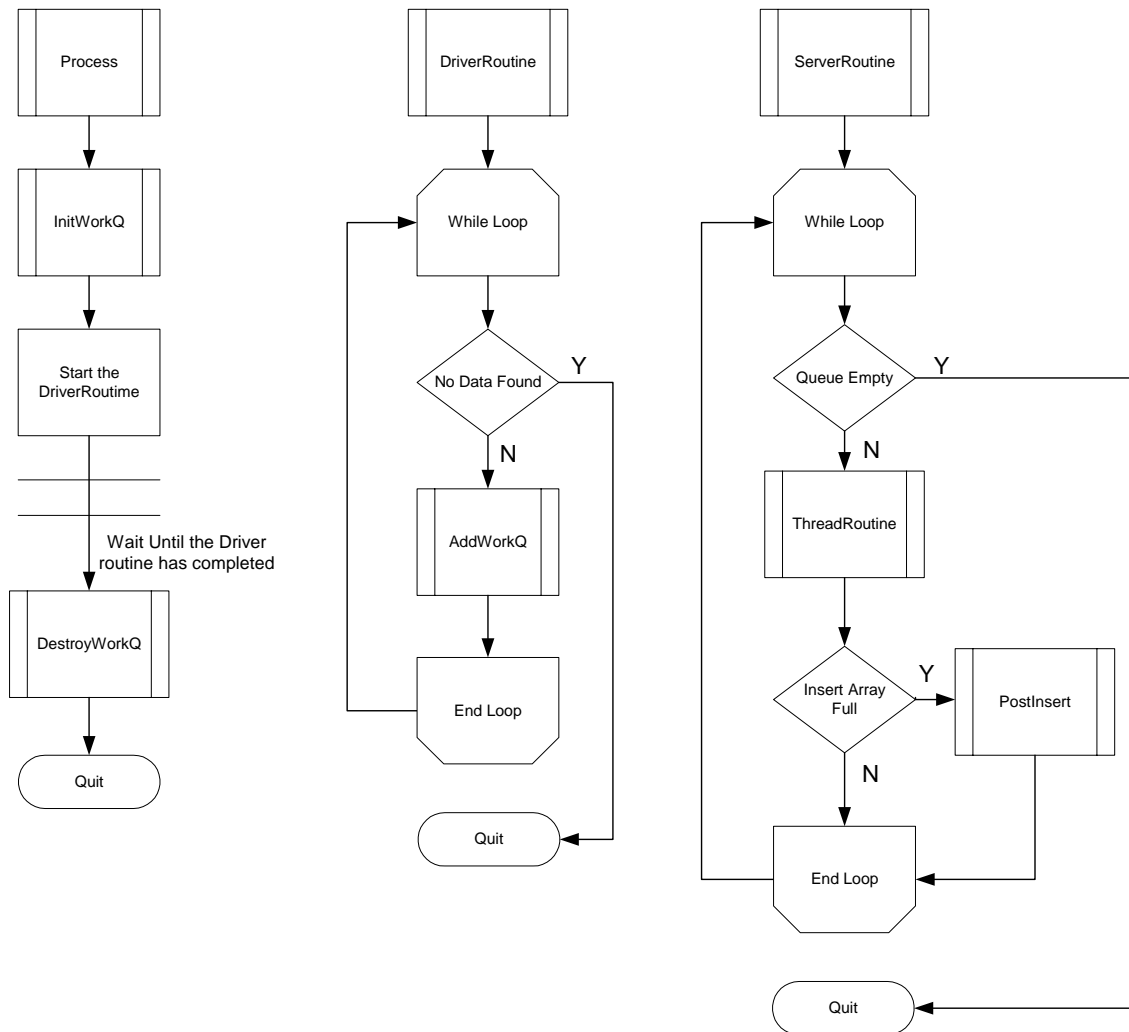
Processing Cycle:	PHASE 3
Scheduling Diagram: need to run before	Prepost (ociroq pre), rplatupd, rpladjf and rpladjs reqext so that all replenishment calculation attributes are up to date. Posupld needs to run before reqext so that all stock information is up to date.
Pre-Processing:	N/A
Post-Processing:	N/A
Threading Scheme:	POSIX threads The restart_control.num_threads will control the number of POSIX threads that are run within ociroq. The batch program ociroq.c itself will only need to be run with one thread.

Restart Recovery

The program processes all items on repl_day for the current day. If the program fails, the rpl_net_inventory_tmp table should be truncated prior to restarting (prepost ociroq pre)

Program Flow

These operations execute in parallel



Shared Modules

GET_REPL_ORDER_QTY_SQL.REPL_METHOD: Stored PL/SQL procedure for calculating the ROQ of an item at a location.

REPLENISHMENT_SQL.GET_STORE_REVIEW_TIME: Stored PL/SQL procedure for calculating the time between scheduled shipments to a store from a warehouse. This time is used by GET_REPL_ORDER_QTY_SQL in its calculations.

OciInitLogon(): C library function that validates the program usage and performs initial environment set-up; including opening the daily log file for writing. It also calls OciConnect().

OciConnect(): C library function that connects to the database and performs some initial environment set-up. This function calls numerous OCI library routines that create the appropriate OCI handles.

OciDisconnct(): C library function that disconnects from the database and free the OCI handles created by the OCISconnect() call.

ReportError(): C library function that calls the OCIErrorGet() function and returns the appropriate error message.

WriteError(): C library function writes the appropriate message to the error file; indicating the type of error encountered and the Oracle Error number and message.

LogMessage(): C library function writes the appropriate message to the log file; indicating start time, end time and time of failure if the program terminated with errors.

RaiseError(): C library function responsible for passing the error code back to the parent process to ensure correct error handling.

Data Structures

repl_info_struct: Holds information fetched from the driving cursor.

GetOltsStruct: Holds the information passed into and returned from the REPL_OLT_SQL.GET_OLTS_AND_REVIEW_TIME procedure.

GetReplStruct: Holds the information passed into and returned from the GET_REPL_ORDER_QTY_SQL.REPL_METHOD procedure.

InsertStruct: Used to buffer the inserts into the repl_net_inventory_tmp table.

DomainStruct: Used to cache forecasting domain information.

Driver_Info: Used by the Driver thread as a container to pass in all the appropriate parameters to the thread routine.

Thread_Info: Used by the Work Queue threads as a container to pass in all the appropriate parameters to the thread routine.

WorkQueue_List: This linked list is used to hold the actual data fetched by the Driver thread to be then consumed by the Work Queue threads.

WorkQueue_Info: Holds all the Work Queue thread control information.

domain_struct: Used to cache forecasting domain information.

Function Level Description

General Controlling Functions

main()

The standard Retek main function, this calls init(), process() and final(), and posts messages to the daily log files.

init()

Fetches system-level global variables and calls other functions to fetch additional global level data; GetNumThreads(), GetStoreCount() and LoadDomainInfo()

Process()

Controls the bulk of the processing. It initializes the Work Queue threads, creates the Driver thread and waits until the Driver thread has completed prior to calling the DestroyWorkQ() and ThreadCleanUp() functions.

final()

The standard Retek final function, this closes down the process and posts messages to the daily logs.

Thread Controlling Functions

InitWorkQ()

Initializes the specific POSIX Pthread library variables used by the Work Queue threads. It then initializes the WorkQueue_Info structure variables and creates the specified number of threads; Each thread calls the ServerRoutine(). The function performs a loop, allocating memory for each threads data structures and connects each thread to the database by calling the OciConnect() library routine. Finally it calls the DefineWorkerStmts() function.

ServerRoutine()

Controls the consumption of the WorkQueue_List. Each Work Queue thread monitors and consumes data from the list until they are instructed to quit or the queue is empty. Initially while the queue is empty the threads poll the queue every 2 seconds checking the status. All thread synchronization is handled by the use of a mutually exclusive lock (mutex). Each node taken from the list is passed to the ThreadRoutine() function.

ThreadRoutine()

Executed by the Work Queue threads; it calls the PL/SQL packages and buffers the result in the InsertStruct. When an individual thread reaches the MAX_INSERT_SIZE the buffer is inserted into the rpl_net_inventory_tmp table.

GetOlts()

Called by the ThreadRoutine(), this function calls the REPL_OLT_SQL.GET_OLTS_AND_REVIEW_TIME PL/SQL package.

GetRepl()

Called by the ThreadRoutine(), this function calls the GET_REPL_ORDER_QTY_SQL.REPL_METHOD PL/SQL package.

DriverRoutine()

Executed by the Driver thread; it's responsible for defining and fetching the driving cursor and adding the batch to the queue. The execution of this function by a thread allows it to run in parallel with the Work Queue threads. The Work Queue threads will start after the first batch has been placed on the WorkQueue_List.

AddWorkQ()

Loads the array fetched by the DriverRoutine() onto the WorkQueue_List. It allocates memory for each node and will continue to load the queue while the number of records on the queue has not exceeded the MAX_QUEUE_SIZE. All thread synchronization is handled by the use of a mutually exclusive lock

(mutex). The function will wait until the queue less than half full prior to recommencing.

DestroyWorkQ()

Waits until all the Work Queue threads have consumed all the data from the list; it then performs some cleanup duties. All thread synchronization is handled by the use of a mutually exclusive lock (mutex).

ThreadCleanUp()

Frees the memory allocated to each threads data structures (including statement handles) and disconnects from the database.

Database DML Handling

PostInsert()

When the Insert buffer reaches the MAX_INSERT_SIZE the array is posted to the database and the work committed.

OCI Statement Functions

DefineDriver()

Performs OCI specific statement set-up; including statement handle preparation, statement handle attribute set-up (pre-fetch size), statement column definition and the array of structure definition (skip size etc.) for the Driving Cursor.

*DefineGetRepl() ***

Performs OCI specific statement set-up; including statement handle allocation, statement handle preparation and statement column binding for the PL/SQL package call GET_REPL_ORDER_QTY_SQL.REPL_METHOD.

*DefineGetOlts() ***

Performs OCI specific statement set-up; including statement handle allocation, statement handle preparation and statement column binding for the PL/SQL package call REPL_OLT_SQL.GET_OLTS_AND_REVIEW_TIME.

*DefineInsert() ***

Performs OCI specific statement set-up; including statement handle preparation, statement handle attribute set-up (pre-fetch size), statement column definition and the array of structure definition (skip size etc.) for the rpl_net_inventory_tmp insert Statement.

**** NOTE:** These functions are called for each Work Queue thread. Each thread will have its own database connection and statement handles.

Database Interaction

The following database tables related to the Net Inventory dialog of RMS and the types of access that will be used by this process: *

Table	Select	Insert	Update	Delete
DOMAIN_CLASS	Y	N	N	N
DOMAIN_DEPT	Y	N	N	N
DOMAIN_SUBCLASS	Y	N	N	N
ITEM_SUPP_COUNTRY	Y	N	N	N
PERIOD	Y	N	N	N
REPL_DAY	Y	N	N	N
REPL_ITEM_LOC	Y	N	N	N
RPL_NET_INVENTORY_TMP	N	Y	N	N

STORE	Y	N	N	N
SYSTEM_OPTIONS	Y	N	N	N
WH	Y	N	N	N
WIN_WH	Y	N	N	N

*NOTE: This list does not include the tables accessed by the PL/SQL package calls executed by this program

I/O Specification

N/A

Technical Issues

N/A