

Contract Maintenance and Purging [cntrmain]

Design Overview

All contracts with statuses not equal to 'Reviewed' or 'Approved' and their associated records will be deleted after a specified number of months has elapsed since their last change in status if there are no existing orders against it.

In the status maintenance function, 'Approved' closed (types A and B) contracts will be set to 'Complete' automatically. Those that are of type C and D will be set to 'Review' status automatically.

Contracts can also be set to 'Complete' manually in the contract header form. Validation on-line will only check outstanding approved orders and no orders exist in the orders tables. In 'Review' status contracts are still active and it is valid to raise new orders against them (manually or automatically). Open contracts (types C and D) will not automatically be set to 'Complete' or 'Cancelled' as this will be a manual action that takes place as part of the review process. The user will only be able to re-approve a contract in review status using the on-line functionality.

When the Batch with Online Users indicator is 'Y', indicating users can be in the system at any given time, checks were set up to determine if any of the records to be deleted or updated are locked by the users. If locked records are encountered, the locked records are written the batch_lock_log table, and will be skipped. If no locks are encountered, processing will continue, and the records subject to deletion or update will be deleted or updated.

This program accesses the following database tables:

TABLE	SELECT	INSERT	UPDATE	DELETE
UNIT_OPTIONS	Yes	No	No	No
PERIOD	Yes	No	No	No
CONTRACT_HEADER	Yes	No	Yes	Yes
CONTRACT_SKU	No	No	No	Yes
CONTRACT_PROD_PLAN	No	No	No	Yes
ORDHEAD	Yes	No	No	No
BATCH_LOCK_LOG	No	Yes	No	Yes
CONTRACT_HEADER_LOCK_TEMP	Yes	Yes	No	Yes

Scheduling Constraints

Processing Cycle: This module will run daily.

Scheduling Diagram: This module needs to be scheduled before the replenishment cycle.

Pre-Processing: None.

Post-Processing: None.

Threading Scheme: None.

Restart Recovery

Contracts pending deletion are read into an array, and all records in contract_cost, contract_detail, and contract_header relating to those contracts are deleted. Transactions are committed after an entire array has been read and deleted. Arrays in this program may need to be smaller than in most other array based restart/recovery programs to control the size of the rollback segment. This is because for each entry in the array records need to be deleted from three tables.

Program Flow

init

- retrieve program variables, including the system_options.btch_w_usr_ind field.
- If the btch_w_usr_ind is 'Y', delete from the batch_lock_log where the program_name is 'cntrmain' and the thread_val is equal to the value fetched by retek_init().

process

- call delete_contracts function
- call reset_inactive function

delete_contracts function

- driving cursor LOOP through contracts valid for deletion on contract_header
- if contract status is C (complete) then
 - validate no valid orders are attached to contract (do not delete contract if an order exists)
 - if the btch_w_usr_ind is 'Y':
 - insert the fetched contract_no and rowid to the contract_header_lock_temp table.
 - call the check_lock().
 - If locked records are encountered, write the locked records to the batch_lock_log table. Skip processing the locked records, and move on to the next array of fetched records.
 - If no locked records are encountered,
 - delete from contract tables
- END LOOP

reset_inactive function

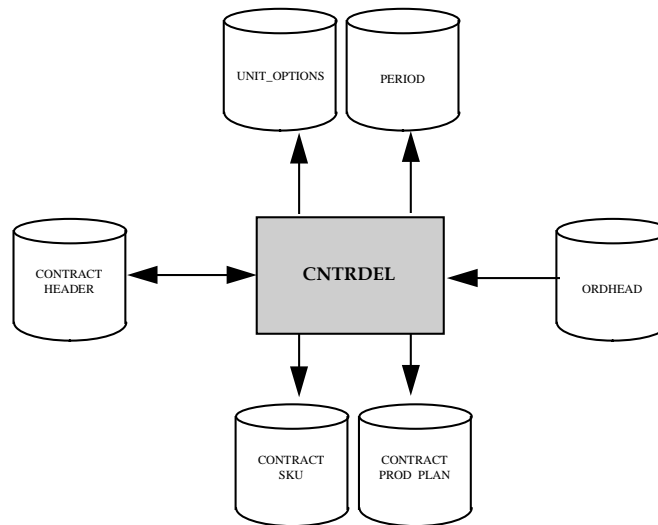
- If the system_options.btch_w_usr_ind is set to 'Y', declare, open and fetch a cursor to fetch the data to be updated from the contract_header table.
- Set the pi_header_only flag to 1 so that only the contract_header table will be checked for locks.
- LOOP through contracts subject to update
 - insert the fetched records into the contract_header_lock_temp table.
 - call the check_lock().
 - If locked records are encountered, write the locked records to the batch_lock_log table. Skip processing the locked records, and move on to the next array of fetched records.
 - If there were no locks encountered,
 - update contract_header
 - where contract type is A or B, approved, and passed the end-date then
 - update contract_header
 - where contract type is C or D, approved, and passed the end-date then
- If the btch_w_usr_ind is set to 'N',

update contract_header
where contract type is A or B, approved, and passed the end-date then
update contract_header
where contract type is C or D, approved, and passed the end-date then

check_lock function

check for locked records on the contract_detail and contract_cost tables, based on the contract_no being processed, if the pi_header_only flag is not equal to 1.
check for locked records on the contract_header table based on the rowid of the contract_no being processed.

Data Flow



Shared Modules

N/A

Function Level Description

include std_err header file and use error handling routines in oracle.pc library.

init()

retrieve system date.

retrieve order_history_months, contract_inactive_months and contract_review_days from unit_options.

retrieve btch_w_usr_ind from system_options.

process():

call delete_contracts function

call reset_inactive function

delete_contracts():

loop

driving cursor on contract_header table with the following where clause: status in ('W', 'S', 'C', 'X', 'D') AND months between status_date and today >= order_history_months AND orders do not exist

exit loop if no rows found.

If the btch_w_usr_ind is 'Y', insert the fetched records into the contract_header_lock_temp table. Call the check_lock() to check for locked records among the data subject to deletion. If locked records are found, write the locked records to the batch_lock_log table, and do not process the array with locked records. If no locked records are found,

delete from the following tables in the specified order using the contract_no selected from the cursor:

- contract_cost
- contract_detail
- contract_header

commit transaction
end loop

reset_inactive():

if the btch_w_usr_ind is 'Y', declare, open and fetch a cursor to select data whose status is 'A' and whose end_date is greater than or equal to the vdate. Set the pi_header_only flag to 1, to ensure that when checking for locks, only the contract_header table will be checked.

LOOP through the fetched records.

 Insert the locked records into the contract_header_lock_temp table.

 Call check_lock().

 If locks are encountered, write the locked records to the batch_lock_log table, then move on to the next array of fetched records.

If no locks are encountered, update contracts where: contract_type is A or B AND approved AND today's date >= end-date and contract_no = current contract_no being processed then set contract_header.status = 'C', and contract_header.status_date and contract_header.completed_date to today's date.

Update contracts where: contract_type is C or D AND approved AND today's date >= end-date and contract_no = current contract_no being processed then set contract_header.status = 'R', contract_header.status_date and contract_header.review_date to today's date, and contract_header.review_id to 'AUTOMATIC'.

If the btch_w_usr_ind is 'N', update contracts where: contract_type is A or B AND approved AND today's date >= end-date then set contract_header.status = 'C', and contract_header.status_date and contract_header.completed_date to today's date.

Update contracts where: contract_type is C or D AND approved AND today's date >= end-date then set contract_header.status = 'R', contract_header.status_date and contract_header.review_date to today's date, and contract_header.review_id to 'AUTOMATIC'.

exception handling: when SQL_ERROR_FOUND, WRITE_ERROR using
SQLCODE and return (-1).

end execution block.

check_lock():

if the pi_header_only flag is not equal to 1:

 check for locks on the contract_detail and contract_cost tables based on the
 contract_no being processed.

 check for locks on the contract_header table based on the rowid of the
 contract_no being processed.

I/O Specification

N/A