

Batch Detail Design

I. Functional Area

Price Changes

II. Module Affected

Postpctranex.pc - New

III. Design Overview

This batch module is responsible for data maintenance tasks that are necessary after running pctranex.

Processed records in PRICE_BATCH_TRAN drive this program. The driving cursor pulls data from the price_batch_tran table representing price changes that will become active the next day. If the Batch with Online Users indicator is set to 'Y', meaning that users can be in the system at any given time, records will be checked for locks prior to being purged from the system. If locked records are encountered, those records must be written to the batch_lock_log table, and will not be deleted from the price_batch_tran table. If no locks are encountered, the processed records will be deleted from the table.

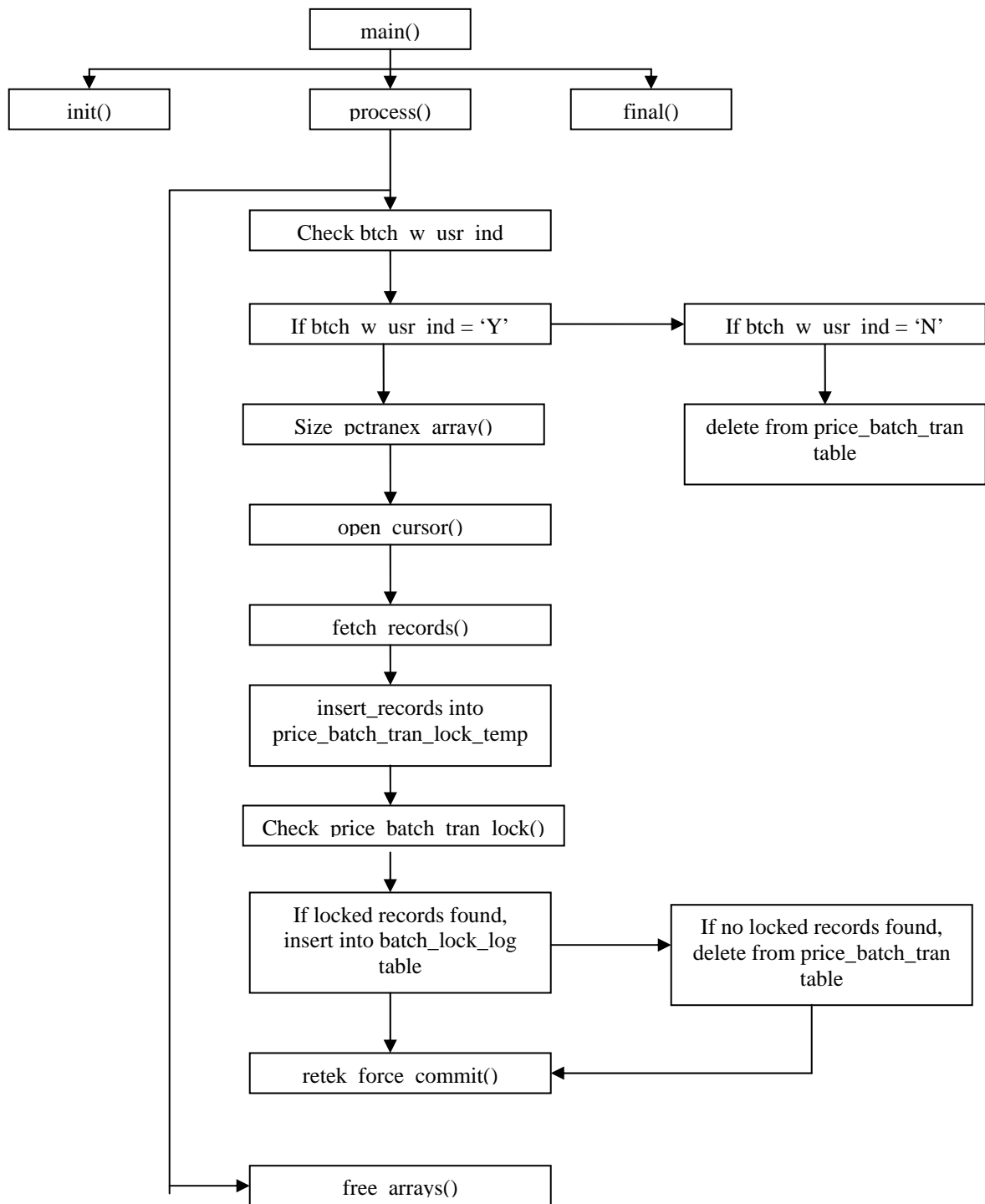
The LUW of this module is a single record from the price_batch_tran table, representing a price change that will take effect the next day.

IV. Stored Procedures / Shared Modules (Maintainability)

N/A



V. Program Flow



VI. Function Level Description

Declare a fetch array struct to hold array-fetched records from the driving cursor.

Declare the driving cursor. The driving cursor fetches primary key and rowid information from the price_batch_tran table, where the effective date is tomorrow, and threading variables match the values fetched by the retek_init().

The cursors should only pick up price_batch_tran records that are related to price_changes going into effect the next day as these are the only ones subject to deletion.

Main(): Standard Retek main function. Validates input parameters, calls init, process and final. Logs appropriate message.

Init(): Standard Retek init function. Calls retek_init() and fetches the vdate and btch_w_usr_ind values from the period and system_options tables. If the btch_w_usr_ind field is 'Y', records are deleted from the batch_lock_log table based on the program_name and thread_val.

Process(): This is the main function that does all the work. It will array fetch the driving cursor until the cursor is empty. If the btch_w_usr_ind is 'N', the program will delete price_batch_tran records which have an effective_date of tomorrow. If the btch_w_usr_ind is 'Y':

- Call open_cursor to open the driving cursor.
- In a while loop (while the no records found indicator is not set):
 - Array fetch records from the driving cursor by calling fetch_cursor().
 - If there are records to be processed, insert the records to be deleted into the price_batch_tran_lock_temp table.
 - Call check_price_batch_tran_lock() to check for locks on the records subject to deletion.
 - If the function returns a value greater than 0, insert the values currently in the price_batch_tran_lock_temp table into the batch_lock_log table.
 - Delete from the price_batch_tran_lock_temp table. If the end of array indicator is set to 1, break from the loop. Otherwise, continue with fetching values.
 - Delete from the price_batch_tran table. The values to be deleted are those without any locks on them.
 - Call retek_force_commit(), with 0 as the argument passed.
- Call free_pctranex_array() to free up the memory used by the array.

Size_arrays(): Sizes the fetch array to the commit size.

Free_arrays(): Frees fetch array.

Check_price_batch_tran_lock(): Checks for locks on the price_batch_tran table based on the rowid.

Final(): Standard Retek final function. Calls retek_close().

VII. Input Specifications

'Table-To-Table'

Select data from:

Table Name	Column Name	Column Type	Transformation
PRICE_BATCH_TRAN	ZONE_GROUP_ID	NUMBER(4)	NONE
PRICE_BATCH_TRAN	OLD_ZONE_ID	NUMBER(10)	NONE
PRICE_BATCH_TRAN	STORE	NUMBER(10)	NONE
PRICE_BATCH_TRAN	NEW_ZONE_ID	NUMBER(10)	NONE
PRICE_BATCH_TRAN	ROWID	ROWID	NONE



PRICE_BATCH_TRAN_LOCK_TEMP	ZONE_GROUP_ID	NUMBER(4)	NONE
PRICE_BATCH_TRAN_LOCK_TEMP	OLD_ZONE_ID	NUMBER(10)	NONE
PRICE_BATCH_TRAN_LOCK_TEMP	STORE	NUMBER(10)	NONE
PRICE_BATCH_TRAN_LOCK_TEMP	NEW_ZONE_ID	NUMBER(10)	NONE
PRICE_BATCH_TRAN_LOCK_TEMP	ROW_ID	ROWID	NONE

VIII. Output Specifications

'Table-To-Table'

Delete from:

Table Name	Column Name	Column Type	Transformation
PRICE_BATCH_TRAN	ZONE_GROUP_ID	NUMBER(4)	NONE
PRICE_BATCH_TRAN	OLD_ZONE_ID	NUMBER(10)	NONE
PRICE_BATCH_TRAN	STORE	NUMBER(10)	NONE
PRICE_BATCH_TRAN	NEW_ZONE_ID	NUMBER(10)	NONE
PRICE_BATCH_TRAN	ROWID	ROWID	NONE
PRICE_BATCH_TRAN_LOCK_TEMP	ZONE_GROUP_ID	NUMBER(4)	NONE
PRICE_BATCH_TRAN_LOCK_TEMP	OLD_ZONE_ID	NUMBER(10)	NONE
PRICE_BATCH_TRAN_LOCK_TEMP	STORE	NUMBER(10)	NONE
PRICE_BATCH_TRAN_LOCK_TEMP	NEW_ZONE_ID	NUMBER(10)	NONE
PRICE_BATCH_TRAN_LOCK_TEMP	ROW_ID	ROWID	NONE
BATCH_LOCK_LOG	PROGRAM_NAME	VARCHAR2(25)	NONE
BATCH_LOCK_LOG	TABLE_NAME	VARCHAR2(32)	NONE
BATCH_LOCK_LOG	KEY_VALUE1	VARCHAR2(25)	NONE
BATCH_LOCK_LOG	KEY_VALUE2	VARCHAR2(25)	NONE
BATCH_LOCK_LOG	KEY_VALUE3	VARCHAR2(25)	NONE
BATCH_LOCK_LOG	KEY_VALUE4	VARCHAR2(25)	NONE
BATCH_LOCK_LOG	KEY_VALUE5	VARCHAR2(25)	NONE
BATCH_LOCK_LOG	KEY_ROWID	ROWID	NONE
BATCH_LOCK_LOG	LOCKED_DATE	DATE	NONE
BATCH_LOCK_LOG	THREAD_VAL	NUMBER(10)	NONE

Update data on:

Table Name	Column Name	Column Type	Transformation

Insert into:

Table Name	Column Name	Column Type	Transformation
PRICE_BATCH_TRAN_LOCK_TEMP	ZONE_GROUP_ID	NUMBER(4)	NONE
PRICE_BATCH_TRAN_LOCK_TEMP	OLD_ZONE_ID	NUMBER(10)	NONE
PRICE_BATCH_TRAN_LOCK_TEMP	STORE	NUMBER(10)	NONE
PRICE_BATCH_TRAN_LOCK_TEMP	NEW_ZONE_ID	NUMBER(10)	NONE
PRICE_BATCH_TRAN_LOCK_TEMP	ROW_ID	ROWID	NONE
BATCH_LOCK_LOG	PROGRAM_NAME	VARCHAR2(25)	NONE
BATCH_LOCK_LOG	TABLE_NAME	VARCHAR2(32)	NONE
BATCH_LOCK_LOG	KEY_VALUE1	VARCHAR2(25)	NONE
BATCH_LOCK_LOG	KEY_VALUE2	VARCHAR2(25)	NONE
BATCH_LOCK_LOG	KEY_VALUE3	VARCHAR2(25)	NONE



BATCH_LOCK_LOG	KEY_VALUE4	VARCHAR2(25)	NONE
BATCH_LOCK_LOG	KEY_VALUE5	VARCHAR2(25)	NONE
BATCH_LOCK_LOG	KEY_ROWID	ROWID	NONE
BATCH_LOCK_LOG	LOCKED_DATE	DATE	NONE
BATCH_LOCK_LOG	THREAD_VAL	NUMBER(10)	NONE

IX. Scheduling Considerations

This module must be run after pctranex in the pricing batch cycle.

This module is multi-threaded by zone_group_id.

X. Locking Strategy

Solution # 1 – Bulk Locking

XI. Restart/Recovery

This program has inherent restart recovery because it deletes records from the price_batch_tran table. Processed records will no longer be picked up because they have already been deleted. This program is threaded by zone_group_id.

XII. Performance Considerations

N/A

XIII. Security Considerations

N/A

XIV. Unit Test Considerations

When program is tested, tester may need to run pctranex prior to running this program. This will imitate the way the batch cycle runs.

See program's UTP for further instructions.

XV. Design Assumptions

Background: The pctranex program updates price_zone_group_store and item_loc records based on the price changes that were fetched from the price_batch_tran table which are set to go into effect the next day. It is assumed that after the pctranex program runs, all the records with an effective date of tomorrow have already been processed and are ready for deletion.



XVI. Outstanding Design Issues

N/A

Issue Description	Priority	Resolution

XVII. Approval and Distribution

The detailed design should be approved by:

Title	Name
Design Lead	

The detailed design should be distributed to:

Title	Name
Quality Control Lead	

XVIII. Appendix

