SAIT Southern Alberta Institute of Technology

CPRG 307

Assignment 2 (Modules 1 – 5, 9 - 10)

Mark: _____ / 13

Problem

This assignment is based on the We Keep It Storage (WKIS) Company's system, an accounting system. Transactions (this is a double-entry accounting system) will be taken from a holding table (NEW_TRANSACTIONS) and inserted into the TRANSACTION_DETAIL and
TRANSACTION_HISTORY tables. At the same time, the appropriate account balance will be updated in the ACCOUNT table. This system follows a double entry accounting system with the accounting rules presented in class.

1. Ensure that you have created the WKIS database tables successfully. These files for this assignment are provided in Module 9 on D2L.

2. Design, write a PL/SQL program, and thoroughly test the coded solution using the information outlined above and the guidelines / restrictions below.

    a. You can assume that every transaction number is unique (there will be no duplicates) for each "transaction". Remember that a transaction is a unit – will be made up of more than one row. All rows that represent a single transaction will have the same transactional history information (TRANSACTION_NUMBER, TRANSACTION_DATE, DESCRIPTION).

    b. Using two nested cursors would make this problem easier to solve. See Module 4 slides and the Module 4 exercise for an example. You do not have to use this method in your solution, this is simply a hint and suggestion for an easier solution.

c. Another hint, get the overall structure of your program working with clean data first. Then start adding error checking and functionality.

d. As long as the debits equal the credits in each transaction, you can assume that the accounting equation for each transaction holds true. For this program, you do **not** have to validate the accounting equation.

e. After a transaction has been successfully processed, remove it from the NEW_TRANSACTIONS table. Transactions that produce an error should remain in the NEW_TRANSACTIONS table.

f. An error in one transaction should not prevent the processing of other transactions (i.e. make sure you do not leave the main looping structure when an error occurs).

g. Only the first error in a transaction should be recorded in the error log table (i.e. a specific transaction number should only appear once in the error log table). If the error is a missing transaction number, a single entry can be recorded in the error log table for all rows missing a transaction number or can have an entry recorded for each row missing a transaction number.

h. When your instructor evaluates your program, only errors that appear in the error log table will be evaluated. If you only print the error to the screen, your instructor will consider this error as not being caught. For debugging you can have an error written to the error log table AND print to the screen, this is fine.

i. All required tables for this assignment, including the error log, are created with the provided scripts. **Do not** create any additional tables or modify the existing tables (structure or constraints).

j. Do **not** use a table of records, or any other type of array, in your solution to this problem. We do not cover these in this course so if you do not know what they are that is fine. **Note: Record data structures are okay; a table of records is different.**

k. SELECT INTO cannot be performed against the NEW_TRANSACTIONS table. A SELECT on NEW_TRANSACTIONS can only be performed by an explicit cursor (use your cursor for any needed values from this table).

l. The solution must be done with **one** anonymous block (multiple embedded blocks are fine as these are not considered separate anonymous blocks). If multiple anonymous blocks are submitted, only the first one in the script will be evaluated by your instructor.

m. Stored programs **cannot** be used.

n. **Do not use** GOTOs, EXITs, or SAVEPOINTs. CONTINUEs can be used if done appropriately (do not use as you would a *break* in Java).

o. There should be **no** hard coding of values anywhere in your code except for the transaction type ('C', 'D') which should be hard coded only in the DECLARE section.

3. Your program should handle all exceptions and write the transactional history information of the transaction that caused the error as well as the error message to the WKIS_ERROR_LOG table.

- Error messages must be descriptive.

- Specific errors that must be caught:

a. Missing transaction number (NULL transaction number)
b. Debits and credits are not equal
c. Invalid account number
d. Negative value given for a transaction amount
e. Invalid transaction type

- All other errors (outside of those listed above) would be considered unanticipated errors. These should be caught as well as we do not want our program to crash. ○ For these unanticipated errors:

The error message would be the system generated error message as we cannot provide a customized descriptive error for this type of problem.

For our purposes, testing does not need to be performed for this type of error. Your instructor will simply be looking for the code to exist that would handle unanticipated errors.

- Your instructor will provide you with two testing data sets.
  - Clean data
  - Mix of clean and erroneous data

- Your instructor will evaluate your program with a **different** set of data.
  - This means that your program should be able to work with any data.
  - Your instructor's data set will be evaluating everything outlined in the evaluation section below, so there will be no surprises. Erroneous data will only be checking for those errors specified above.
  - Do not code to the data sets, code to the problem. The test data is simply there to help you confirm your code is working as it should.
  - It is highly recommended you also create additional test data to ensure your program works regardless of the data provided.