The Smartcard Financial Settlement System

Contents

Introduction

Smartcard System has been used to allow merchants and consumers to pay or receive electronic cash conveniently. Currently Smartcard System is used by a various of merchants with a various of terminals like parking meters vending machines, payphones , contact less Smartcard readers and so on. Each day the Smartcard Financial Settlement System (FSS) will download all the transactions that merchants have uploaded, and then will settle all the settlement for merchants as long as the settlement is greater than the daily minimum settlement, otherwise those settlement will be settled by the end of each month regardless of the amount. FSS will also produce a daily desk bank file to be electronically sent to designated banking or organization and a daily banking summary report for Smartcard Company's business department. FSS also needs to produce a fraud report for the security department to identify all the transactions that are suspected as fraud.

Task

The tasks are:

1. Create the daily settlement system

2. Design the FSS system to produce a desk bank file, a daily settlement report and a fraud report with each successful run.

3. Maintain a Run Table so that the production support personnel can use this table to monitor and maintain the system in the event of system failure.

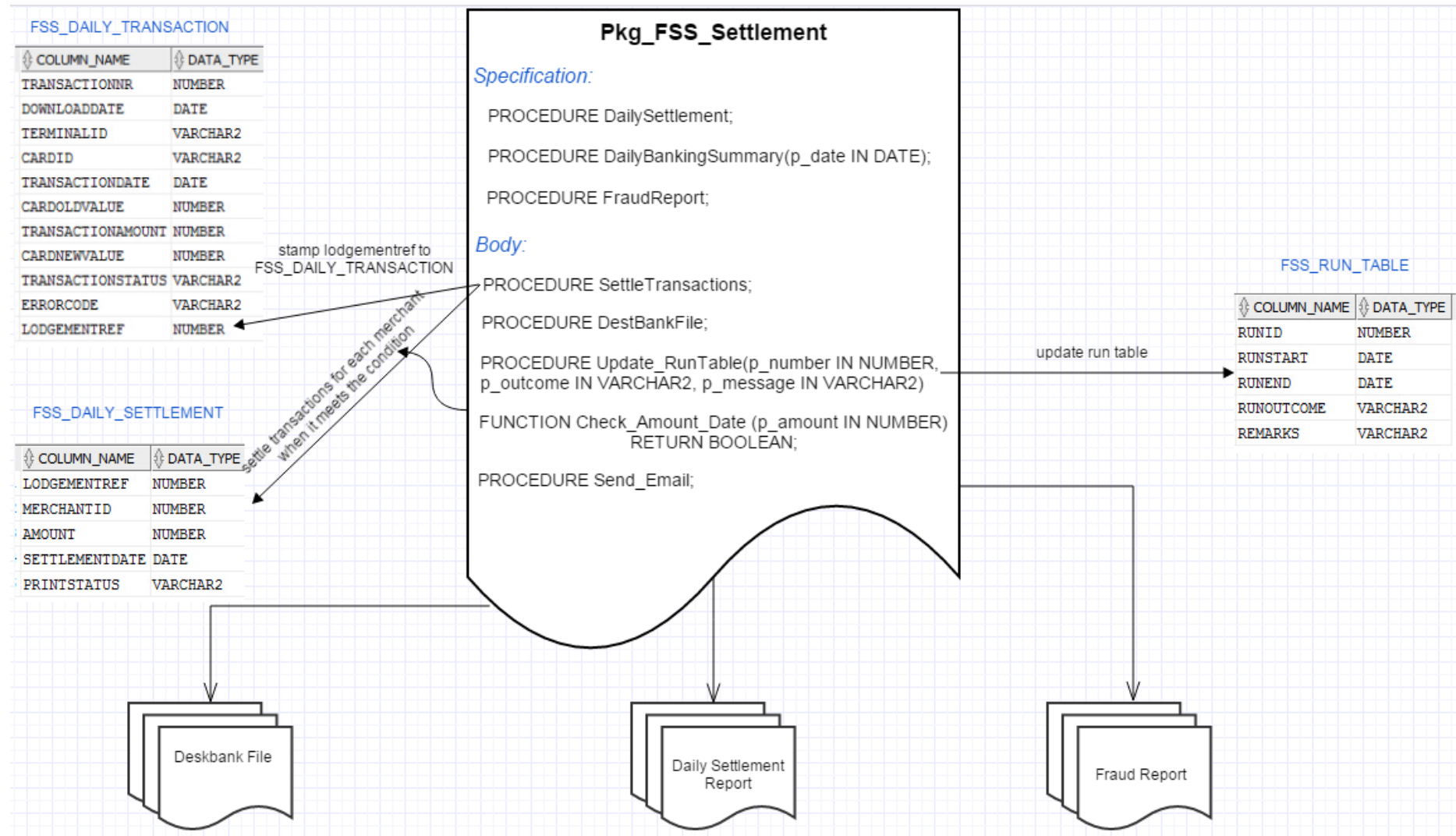## How to effectively run the program

The program is stored in a single package called Pkg_FSS_Settlement. There are comments in each procedure and function in the package to describe purposes in order to help developers understand the variable names that were used and what do these names stand for and the design of the solution. There are also comments highlighting business rules contained in the code. To run the program, simply execute the following anonymous block:

BEGIN

Pkg_FSS_Settlement.DailySettlement;

END;

Design of the Solution

FSS_DAILY_TRANSACTION

COLUMN_NAME	DATA_TYPE
TRANSACTIONNR	NUMBER
DOWNLOADDATE	DATE
TERMINALID	VARCHAR2
CARDID	VARCHAR2
TRANSACTIONDATE	DATE
CARDOLDVALUE	NUMBER
TRANSACTIONAMOUNT	NUMBER
CARDNEWVALUE	NUMBER
TRANSACTIONSTATUS	VARCHAR2
ERRORCODE	VARCHAR2
LODGEMENTREF	NUMBER

FSS_DAILY_SETTLEMENT

COLUMN_NAME	DATA_TYPE
LODGEMENTREF	NUMBER
MERCHANTID	NUMBER
AMOUNT	NUMBER
SETTLEMENTDATE	DATE
PRINTSTATUS	VARCHAR2

stamp lodgementref to
FSS_DAILY_TRANSACTION

settle transactions for each merchant
when it meets the condition

Pkg_FSS_Settlement

Specification:

  PROCEDURE DailySettlement;

  PROCEDURE DailyBankingSummary(p_date IN DATE);

  PROCEDURE FraudReport;

Body:

  PROCEDURE SettleTransactions;

  PROCEDURE DestBankFile;

  PROCEDURE Update_RunTable(p_number IN NUMBER,
  p_outcome IN VARCHAR2, p_message IN VARCHAR2)

  FUNCTION Check_Amount_Date (p_amount IN NUMBER)
           RETURN BOOLEAN;

  PROCEDURE Send_Email;

update run table

FSS_RUN_TABLE

COLUMN_NAME	DATA_TYPE
RUNID	NUMBER
RUNSTART	DATE
RUNEND	DATE
RUNOUTCOME	VARCHAR2
REMARKS	VARCHAR2

Deskbank File

Daily Settlement Report

Fraud Report

Functionality:

There are two sequences which are seqid_key and runid_key. seqid_key is used for generating primary key for FSS_Daily_Settlement and runid_key is used for generating primary key for FSS_Run_Table.

There are two common functions that can be used by any person, they are:

FUNCTION f_centre(p_text VARCHAR2, p_pageWidth NUMBER) RETURN VARCHAR2 IS

This function takes two parameters, the first parameter is the text that needs to be centre in a line and the second parameter is the width of a page, which can be used for LPADing space in p_text which will be returned to caller procedures.

FUNCTION get_code_value(p_kind VARCHAR2, p_code VARCHAR2) RETURN VARCHAR2 IS

This function takes two parameters to get the value of PARAMETER table. In this task, p_kind is `'EMAIL_ADDRESS'` and p_code is `'ASS2_RECIPIENT'`

Pkg_FSS_Settlement:

PROCEDURE DailySettlement

This is the trigger procedure to run the program. Firstly it inserts a record in the FSS_Run _Table to record its current run information and then it looks for the previous run record in the FSS_Run _Table to check if the program had run before or it is still running. If the program has not been run during the day, then it will copy all the new transactions from FSS_TRANSACTIONS table to FSS_DAILY_TRANSACTION table. The program will be aborted if last run has not been finished or it has ran before.

FUNCTION Check_Amount_Date(p_amount IN NUMBER) RETURN BOOLEAN:

This function takes a parameter of number type which is the amount of settlement passed from SettleTransactions procedure. It will compare the settlement with the minimum settlement amount in FSS_Reference. It will return true to the caller procedure if the value is greater than the minimum settlement or the sysdate is the last day of the month, so that SettleTransactions procedure can decide whether it should settle a settlement for a merchant.

PROCEDURE SettleTransactions (p_newSettlement IN OUT NUMBER)

This procedure takes the p_newSettlement as a parameter that represents the number of the total settlements, it is in both IN and OUT as the caller procedure will need to use this parameter to update the FSS_Run_Table by indicating how many settlement are settled in a day. A cursor called c_merchant_Total is used for selecting each merchant with their total settlement amount using sum fuction and group by statement, then the total amount will be passed to the function Check_Amount_Date(p_amount IN NUMBER) to determine whether it need to be settled. It will insert a settlement record whose primary key is lodgementref in FSS_DAILY_SETTLEMENT if the amount needs to be settled. To generate lodgementref which is a date concatenated with a sequence number generated by seq_id. One important thing is that we need to trace back all transactions in FSS_Transactions that belongs to this settlement to stamp lodgementref into these transaction. In this way it will not re-settle transactions that have been settled before.

PROCEDURE Update_RunTable(p_number IN NUMBER, p_outcome IN VARCHAR2, p_message IN VARCHAR2):

This procedure takes three parameter to update FSS_Run_Table or insert a new record into it. The logic is : if p_number is not null, then the program knows it needs to insert a new record into the table. p_number generated by run_id sequence in the main procedure will be used as the primary key for this record and sysdate will be the runstart. If p_outcome is passed as "SUCCESS", it will update the runend to sysdate and put message into remarks use by using CURRVAL value of run_id to locate the primary key of the record. Same process applies to p_outcome when it is "FAIL" as well.

PROCEDURE Send_Email;

This procedure is used to send an email with attachment of Daily Settlement Report to a nominated person.

PROCEDURE DailyBankingSummary(p_date IN DATE):

This procedure takes a date type parameter which can be an input from authorised persons who want to print a summary report for a specific period of time. The DailySettlement procedure will pass sysdate to it so that the program can automatically print a daily settlement report with each run.

PROCEDURE DestBankFile;

This procedure is used to print the daily deskbank file.

PROCEDURE FraudReport;

This procedure is used to print fraud reports. A cursor in cursor is used to get fraud transactions. The basic idea is:

Order each card's cardid. Use one cursor to get each cardid as the parameter of next cursor which is used to select all transactions that belong to this cardid. Then this procedure will compare the difference between the old value of the new transaction and new value of previous transaction in a loop as the diagram below shows:

TRANSACTIONNR	TERMINALID	CARDID	TRANSACTIONDATE	CARDOLDVALUE	CARDNEWVALUE
88028	0061001320	61022004000002179	06/MAY/15	3041	2494
88029	0031000040	61022004000002179	06/MAY/15	2494	2111
88030	0022000060	61022004000002179	06/MAY/15	2111	1941
88031	0031000030	61022004000002179	06/MAY/15	1941	1586
88032	0051001800	61022004000002179	06/MAY/15	1586	1536
88033	0022000050	61022004000002179	06/MAY/15	1536	1350
88034	0061001340	61022004000002179	06/MAY/15	1350	991
88035	0051000800	61022004000002179	06/MAY/15	991	941
88036	0022000070	61022004000002179	06/MAY/15	941	762
88037	0051000600	61022004000002179	06/MAY/15	762	712
88038	0061001300	61022004000002179	06/MAY/15	712	137
88039	0051001500	61022004000002179	06/MAY/15	137	87
88040	0051001000	61022004000002179	06/MAY/15	87	37
88319	0771000010	61022004000002179	07/MAY/15	37	32
92800	0771000010	61022004000002179	16/MAY/15	100000	9500