MERGE 2 LISTS

LOOP  ITEMS one by one

BEGIN

        IF IT IS THE FIRST TIME OR THE INFORMATION FOR CT / SERVER / CUSTOMER IS DIFFERENT

               BEGIN

                           INSERT / UPDATE **SCHEDULE**  AND BRING BACK THE SCHEDULE ID

                           (USUALLY THIS WILL BE DONE 1 TIME IN THE WHOLE PROCESS, FILE IS NOT MIXING MULTIPLE SCHEDULES, ONLY 1 UNTIL NOW )

                 END

        INSERT / UPDATE  **PLAN\_SHRINK** FOR EACH ITEM

        (HERE IS WHERE THE PROCESS IS TAKING A LOT OF TIME)

END LOOP

//MERGE 2 LISTS INTO 1

List<ScheduleLoadEntity> schedulel = GetExcel();

List<ScheduleLoadEntity> informacion = new DashboardBO().GetDIM\_IEX\_CT();

IEnumerable<ScheduleLoadEntity> scheduleDetailsList2 = (from excel in schedulel

from info in informacion.Where(info =>

info.APP\_CODE == excel.APP\_CODE

&& info.CUSTOMER\_ID == excel.CUSTOMER\_ID

&& info.CT\_ID == excel.CT\_ID)

select new ScheduleLoadEntity

{

APP\_CODE = excel.APP\_CODE,

CUSTOMER\_ID = excel.CUSTOMER\_ID,

CT\_ID = excel.CT\_ID,

IEX\_CT\_SK = info.IEX\_CT\_SK,

ECP\_LOB\_ID = info.ECP\_LOB\_ID,

SCHEDULE\_RELEASE\_DATE = excel.SCHEDULE\_RELEASE\_DATE

}).Distinct();

if (scheduleDetailsList2.Count() == 0)

{

return "Data could not be referred in database, check values in excel file for Server, Customer and CT\_ID ";

}

int LOB = 0;

int SK = 0;

int cust\_id = 0;

int CT\_ID = 0;

string server = "";

int schedule\_id = 0;

int conteo = 0;

int conteo2 = 0;

DataSet ds = new DataSet();

ds.Tables.Add("DS");

ds.Tables["DS"].Columns.Add("IEX\_SERVER");

ds.Tables["DS"].Columns.Add("CUSTOMER\_ID");

ds.Tables["DS"].Columns.Add("CT\_ID");

ds.Tables["DS"].Columns.Add("SCHEDULE\_RELEASE\_DATE");

ds.Tables["DS"].Columns.Add("SCHEDULE\_DATE");

ds.Tables["DS"].Columns.Add("INTERVAL");

ds.Tables["DS"].Columns.Add("PLAN\_SHRINK");

//foreach (DataRow items in dtExcel.Rows)

foreach (var row in scheduleDetailsList2)

{

try

{

string p\_IEX\_SERVER = row.APP\_CODE;

int p\_CUSTOMER\_ID = row.CUSTOMER\_ID;

int p\_CT\_ID = row.CT\_ID;

DateTime p\_SCHEDULE\_RELEASE\_DATE = row.SCHEDULE\_RELEASE\_DATE;

DateTime p\_SCHEDULE\_DATE = row.SCHEDULE\_DATE;

int p\_INTERVAL = row.INTERVAL;

int p\_PLAN\_SHRINK = row.PLAN\_SHRINK;

if (server != p\_IEX\_SERVER || CT\_ID != p\_CT\_ID || cust\_id != p\_CUSTOMER\_ID)

{

server = p\_IEX\_SERVER;

CT\_ID = p\_CT\_ID;

cust\_id = p\_CUSTOMER\_ID;

//MASTER

schedule\_id = new DashboardBO().Load\_Schedule(LOB, p\_CT\_ID,

p\_SCHEDULE\_RELEASE\_DATE,

p\_IEX\_SERVER, p\_CUSTOMER\_ID, SK);

bool ss2 = new DashboardBO().Load\_Schedule\_Plans(schedule\_id, p\_INTERVAL, p\_PLAN\_SHRINK, p\_SCHEDULE\_DATE);

conteo++;

}

catch(Exception ex)

{

conteo2++;

DataRow dr = ds.Tables[0].NewRow();

dr[0] = row.APP\_CODE;

dr[1] = row.CUSTOMER\_ID;

dr[2] = row.CT\_ID;

dr[3] = row.SCHEDULE\_RELEASE\_DATE;

dr[4] = row.SCHEDULE\_DATE;

dr[5] = row.INTERVAL;

dr[6] = row.INTERVAL;

//for (int j = 0; j < 7;j++ )

//{

// dr[j] = row[j].ToString();

//}

ds.Tables[0].Rows.Add(dr);

}

}

if (conteo2>0)

{

ShowExcel(ds.Tables[0]);

}

string fallo = ", failed records = "+ conteo2.ToString();

if (conteo2 == 0) fallo = "";

return "Loaded "+ conteo + " records " + fallo;

}

= = = = = = = = = = = = = = = = = = = = = = =

**ANOTHER WAY**

MERGE 2 LISTS

GET FROM MERGE LIST THE GROUP, A NEW LIST FOR HEADER

LOOP  ITEMS in HEADER GROUPED

BEGIN

GET DETAIL ACCORDING ITEM IN LOOP

INSERT/UPDATE HEADER

INSERT/UDPATE DETAIL (SENDING THE LIST, NOT ONE BY ONE)

END LOOP

List<ScheduleLoadEntity> schedulel = GetExcel();

List<ScheduleLoadEntity> informacion = new DashboardBO().GetDIM\_IEX\_CT();

//important step: merge the two lists excel and information

var scheduleDetailsList2 = (from excel in schedulel

join info in informacion

on new {excel.IEX\_SERVER, excel.CUSTOMER\_ID, excel.CT\_ID} equals

new {info.IEX\_SERVER, info.CUSTOMER\_ID, info.CT\_ID} into ServerInfo

select new ScheduleLoadEntity

{

IEX\_SERVER = excel.IEX\_SERVER,

CUSTOMER\_ID = excel.CUSTOMER\_ID,

CT\_ID = excel.CT\_ID,

IEX\_CT\_SK = ServerInfo.First().IEX\_CT\_SK,

ECP\_LOB\_ID = ServerInfo.First().ECP\_LOB\_ID,

SCHEDULE\_RELEASE\_DATE = excel.SCHEDULE\_RELEASE\_DATE,

SCHEDULE\_DATE = excel.SCHEDULE\_DATE,

INTERVAL = excel.INTERVAL,

PLAN\_SHRINK = excel.PLAN\_SHRINK

}).Where(e=>e.ECP\_LOB\_ID > 0).Distinct();

if (scheduleDetailsList2.Count() == 0)

{

return "Data could not be referred in database, check values in excel file for Server, Customer and CT\_ID ";

}

// SEGUNDA LISTA, SAQUE LAS AGRUPACIONES, EL HEADER

var HeaderGrouped = scheduleDetailsList2.GroupBy(info => new

{

IEX\_SERVER = info.IEX\_SERVER,

CUSTOMER\_ID = info.CUSTOMER\_ID,

CT\_ID = info.CT\_ID,

});

bool success = false;

List<ScheduleLoadBase> failedRecords = new List<ScheduleLoadBase>();

foreach (var row in HeaderGrouped)

{

IList<ScheduleLoadEntity> **rowsToUpload** = null;

try

{

// cargue en rowsToUpload conforme a cada grupo

ScheduleLoadEntity entity = row.First();

**rowsToUpload** = row.ToList(); // de tipo IList<ScheduleLoadEntity

// Insert o Update Header/Master

int schedule\_id = new DashboardBO().Load\_Schedule(entity.ECP\_LOB\_ID, row.Key.CT\_ID,

entity.SCHEDULE\_RELEASE\_DATE, row.Key.IEX\_SERVER, row.Key.CUSTOMER\_ID, entity.IEX\_CT\_SK);

// insert o update Detail

success = new DashboardBO().Load\_Schedule\_Plans(schedule\_id, **rowsToUpload**);

}

catch(Exception ex)

{

Common.DisplayMessage(this, "error", ExceptionLogging.LogException(ex));

return ex.ToString();

}

if(success == false && rowsToUpload != null)

{

failedRecords.AddRange(rowsToUpload.Select(rec => new ScheduleLoadBase

{

IEX\_SERVER = rec.IEX\_SERVER,

CUSTOMER\_ID = rec.CUSTOMER\_ID,

CT\_ID = rec.CT\_ID,

SCHEDULE\_RELEASE\_DATE = rec.SCHEDULE\_RELEASE\_DATE,

SCHEDULE\_DATE = rec.SCHEDULE\_DATE,

INTERVAL = rec.INTERVAL,

PLAN\_SHRINK = rec.PLAN\_SHRINK

}));

}

}

if (failedRecords.Count > 0)

{

ShowExcel(failedRecords);

}

return "Data Processing Completed";

}

}

FOR HEADER, RETURN NEW SEQUENTIAL VALUE

database.AddInParameter(dbCommand, "p\_CT\_ID", OracleDbType.Int32, p\_CT\_ID);

database.AddInParameter(dbCommand, "p\_CUSTOMER\_ID", OracleDbType.Int32, p\_CUSTOMER\_ID);

database.**AddOutParameter**(dbCommand, "p\_SCHEDULE\_ID", OracleDbType.Int32, 9);

database.AddInParameter(dbCommand, "p\_SCHEDULE\_START\_DATE", OracleDbType.Date, p\_SCHEDULE\_START\_DATE);

database.AddOutParameter(dbCommand, "p\_errmsg", OracleDbType.Varchar2, 4000);

database.ExecuteNonQuery(dbCommand, connection);

p\_errmsg = database.GetParameterValue(dbCommand, "p\_errmsg").ToString();

if (p\_errmsg.Trim().Length == 0 || p\_errmsg == "null")

{

**scheduleID = ((OracleDecimal)database.GetParameterValue(dbCommand, "p\_SCHEDULE\_ID")).Value.ToInt();**

}

else

{

throw new BusinessLogicException(p\_errmsg.ToString());

}

PROCEDURE Load\_Schedules2(p\_LOB\_ID in NUMERIC,

p\_CT\_ID in NUMERIC,

p\_CUSTOMER\_ID in NUMERIC,

**p\_SCHEDULE\_ID** OUT NUMERIC,

p\_SCHEDULE\_START\_DATE in DATE,

p\_SCHEDULE\_RELEASE\_DATE in DATE,

p\_errmsg OUT VARCHAR2) IS

BEGIN

Begin

SELECT schedule\_id into **P\_SCHEDULE\_ID**

FROM CC\_CT\_SCHEDULE

WHERE CT\_ID = p\_CT\_ID

AND CUSTOMER\_ID = p\_CUSTOMER\_ID

AND TRUNC(SCHEDULE\_START\_DATE) = trunc(p\_SCHEDULE\_START\_DATE);

exception

WHEN OTHERS THEN

null;

end;

IF (p\_SCHEDULE\_ID > 0) THEN

UPDATE CC\_CT\_SCHEDULE

SET SCHEDULE\_RELEASE\_DATE = p\_SCHEDULE\_RELEASE\_DATE,

UPDATED\_ON = SYSDATE,

CUSTOMER\_ID = p\_CUSTOMER\_ID

WHERE SCHEDULE\_ID = p\_SCHEDULE\_ID;

ELSE

select CC\_CT\_SCHEDULE\_ID\_SEQ.NEXTVAL into P\_SCHEDULE\_ID from dual;

INSERT INTO CC\_CT\_SCHEDULE

(

SCHEDULE\_ID,

CT\_ID,

SCHEDULE\_START\_DATE,

CREATED\_ON,

CUSTOMER\_ID )

VALUES

(

P\_SCHEDULE\_ID ,

p\_CT\_ID,

p\_SCHEDULE\_START\_DATE,

SYSDATE,

p\_CUSTOMER\_ID);

END IF;

Exception

WHEN OTHERS THEN

p\_errMsg := SQLERRM;

END Load\_Schedules2;

FOR DETAIL

internal bool Load\_Schedule\_Plans( int schedule\_id, IList<ScheduleLoadEntity> **loadDetails**)

{

bool commit;

StringBuilder sbQuery = new StringBuilder();

sbQuery.AppendLine("**declare** v\_rowCount number; ");

sbQuery.AppendLine("**BEGIN** ");

sbQuery.AppendLine("**SELECT** count(\*) **INTO** v\_rowCount **FROM** CC\_CT\_PLAN\_SHRINK "

+" WHERE INTERVAL = :p\_INTERVAL AND SCHEDULE\_ID = :P\_SCHEDULE\_ID AND schedule\_date = :p\_SCHEDULE\_DATE; ");

sbQuery.AppendLine("IF v\_rowCount > 0 THEN ");

sbQuery.AppendLine("**UPDATE** CC\_CT\_PLAN\_SHRINK SET plan\_shrink = :p\_PLAN\_SHRINK, updated\_on = SYSDATE");

sbQuery.AppendLine("**WHERE** INTERVAL = :p\_INTERVAL AND SCHEDULE\_ID = :P\_SCHEDULE\_ID AND schedule\_date = :p\_SCHEDULE\_DATE;");

sbQuery.AppendLine("**ELSE** ");

sbQuery.AppendLine("**INSERT** **INTO** CC\_CT\_PLAN\_SHRINK (PLAN\_SHRINK\_ID, SCHEDULE\_ID, INTERVAL, PLAN\_SHRINK, CREATED\_ON, SCHEDULE\_DATE)");

sbQuery.AppendLine("**VALUES** (Cc\_ct\_plan\_shrink\_id\_seq.NEXTVAL, :P\_SCHEDULE\_ID, :p\_INTERVAL, :p\_PLAN\_SHRINK, SYSDATE, :p\_SCHEDULE\_DATE);");

sbQuery.AppendLine("END IF;");

sbQuery.Append("END;");

// CREATE 2 ARRAYS, AND ASSIGN DEFAULT VALUES, schedule\_id and created by

int[] scheduleIDArray = new int[loadDetails.Count];

string[] createdByArray = new string[loadDetails.Count];

scheduleIDArray.InitializeAll(schedule\_id);

createdByArray.InitializeAll(CREATED\_BY);

// CREATE ARRAY WITH THE REST OF VALUES

int[] intervalArray = **loadDetails**.Select(rec => rec.INTERVAL).ToArray();

using (OracleCommand dbCommand = database.GetSqlStringCommand(sbQuery.ToString()))

{

dbCommand.ArrayBindCount = loadDetails.Count;

database.AddInParameter(dbCommand, ":p\_SCHEDULE\_ID", OracleDbType.Int32, scheduleIDArray);

database.AddInParameter(dbCommand, ":p\_INTERVAL", OracleDbType.Int32, intervalArray);

database.AddInParameter(dbCommand, ":p\_PLAN\_SHRINK", OracleDbType.Decimal, **loadDetails**.Select(rec => rec.PLAN\_SHRINK).ToArray());

database.AddInParameter(dbCommand, ":p\_SCHEDULE\_DATE", OracleDbType.Date, **loadDetails**.Select(rec => rec.SCHEDULE\_DATE).ToArray());

using (OracleConnection connection = database.CreateConnection())

{

try

{

connection.Open();

database.ExecuteNonQuery(dbCommand, connection);

commit = true;

}

finally

{

database.CloseConnection(connection);

}

}

}

return commit;

}

}