Paper 063-30

Automation Of Code Leads To Vacation For You: Enabling And Disabling Table Constraints In An Oracle® Table With SAS® X Commands

David Steves, SunTrust Bank, Atlanta, Georgia
Denise A Figliozzi, SunTrust Bank, Atlanta, Georgia

ABSTRACT

SAS X Commands are the answer! Loading SAS data into Oracle tables can be executed in a batch process. When loading a large amount of data (that we know is correct) to the Oracle table, it is wise to disable the foreign key constraints for time efficiencies. After the load has completed, the constraints must be re-enabled. This can be accomplished by using the PL/SQL Procedure, CONTROL_FKEYS. But the question remains – how can SAS invoke this PL/SQL code automatically? Again I tell you: SAS X Commands are the answer!

This paper will detail the necessary process of executing the SQL scripts from a SAS batch job. It is always in your own best interest to automate code as fully as possible. This way, you can leave the office and head out for vacation!

INTRODUCTION

Simplicity! Simplicity! Simplicity! Large companies often fail to chant that mantra when creating an IT process. Processes that have become convoluted can be simplified with the use of a SAS X command.

We found ourselves on an ETL project of loading Oracle tables. These tables often contain Referential Integrity (RI) constraints that become disabled during a bulk load. Constraints become disabled so that a bulk load can run smoothly and efficiently, thus populating the Oracle tables.

After the load we had to contact our Oracle Database Administrator to enable constraints. Oftentimes it was difficult to contact the DBA because the tables were loaded during off peak work hours. In an effort to alleviate the role of an Oracle DBA, we found a PL/SQL procedure to enable constraints.

In this paper, we will highlight three components of running a PL/SQL procedure that include the following: the creation of PL/SQL procedure, the testing of the PL/SQL procedure, and finally the submission of the PL/SQL procedure through an X command.

PL/SQL PROCEDURE

Sometimes procedures that you have a need for may have already been created. We have found several procedures that are helpful. Why invent, when someone has painstakingly gone through the process. The CONTROL_FKEYS procedure is a PL/SQL procedure that provides an automated method of enabling and disabling constraints on a specific table.

```
CREATE OR REPLACE PROCEDURE control_fkeys(
  p_table_name  IN user_constraints.table_name%TYPE  ,p_enable_flag IN
NUMBER
 ,p_status      IN OUT NUMERIC)
AUTHID DEFINER IS
  -- constants
  k_enable  user_constraints.status%TYPE := 'ENABLE';
  k_disable user_constraints.status%TYPE := 'DISABLE';
  -- identify fkeys on pkey for given table
  CURSOR id_fkeys (
    c_table_name user_constraints.table_name%TYPE
   ,c_status user_constraints.status%TYPE)
 IS
 /* Note: this select gets ALL FK constraints for the specified table,
both inbound and outbound (parent and child fk constraints. For now, we
want to control ALL the FK constraints against a specified table so use
this select statement.
 */
  SELECT table_name, constraint_name fkey, r_constraint_name
pkey,status
  FROM user_constraints
  WHERE ( constraint_type='R' and table_name = c_table_name)  --outbound
fk constraints
  or
  (   r_constraint_name IN (
         --inbound fk constraints
     SELECT constraint_name
     FROM user_constraints
      WHERE table_name = c_table_name
      AND constraint_type='P')
  )
  AND status!=c_status
  ORDER BY table_name, constraint_name;

  /* Note: this select only gets FK constraints that are inbound, meaning
the specified table contains parent records.
  */
--  SELECT table_name, constraint_name fkey, r_constraint_name pkey,
status
--  FROM user_constraints
--  WHERE constraint_type='R'
--    AND status!=c_status
--    AND r_constraint_name IN (
--     SELECT constraint_name
--      FROM user_constraints
--      WHERE table_name=c_table_name
--      AND constraint_type='P')
--  ORDER BY table_name, constraint_name;

  -- record variables
  rec_id_fkeys id_fkeys%ROWTYPE;
  -- variables
  l_status user_constraints.status%TYPE;
  l_table_name user_constraints.table_name%TYPE;
  l_stmt VARCHAR2(255);
  l_pkey_name user_constraints.constraint_name%TYPE;
BEGIN
  p_status := 0;
  l_table_name := UPPER(p_table_name);
  IF (p_enable_flag = 1) THEN
    l_status := k_enable;
  ELSIF (p_enable_flag = 0) THEN
    l_status := k_disable;
  ELSE
    DBMS_OUTPUT.put_line(
      '-- control_fkeys: enable_flag must be 1 or 0 [' || p_enable_flag
|| ']');
    p_status := 1001;
  END IF;
  IF (p_status = 0) THEN  -- validated enable flag
    -- a primary key for the given table must exist
    SELECT constraint_name
    INTO l_pkey_name
    FROM user_constraints
    WHERE table_name=l_table_name
      AND constraint_type='P';
    DBMS_OUTPUT.put_line(
      '-- control_fkeys: ' || l_status ||
      ' foreign key constraints on table ' || l_table_name ||
      ' whose primary key is ' || l_pkey_name);
    OPEN id_fkeys(l_table_name, l_status || 'D');
    LOOP -- process foreign keys
      FETCH id_fkeys INTO rec_id_fkeys;
      EXIT WHEN id_fkeys%NOTFOUND;

         IF l_status = k_enable THEN
```

```
               l_stmt := 'ALTER TABLE ' || rec_id_fkeys.table_name
                           ||' ENABLE NOVALIDATE CONSTRAINT ' ||
rec_id_fkeys.fkey;
                    DBMS_OUTPUT.put_line(l_stmt);
                           EXECUTE IMMEDIATE l_stmt;
               l_stmt := 'ALTER TABLE ' || rec_id_fkeys.table_name || '
MODIFY '
                           ||' CONSTRAINT ' || rec_id_fkeys.fkey || '
VALIDATE' ;
                    DBMS_OUTPUT.put_line(l_stmt);
                           EXECUTE IMMEDIATE l_stmt;
             ELSE
               l_stmt := 'ALTER TABLE ' || rec_id_fkeys.table_name
                           ||' DISABLE CONSTRAINT ' || rec_id_fkeys.fkey;
                    DBMS_OUTPUT.put_line(l_stmt);
                           EXECUTE IMMEDIATE l_stmt;
             END IF;

    END LOOP; -- process foreign keys
    IF (id_fkeys%ROWCOUNT = 0) THEN  -- no fkeys found that weren't
enabled/disabled
      DBMS_OUTPUT.put_line(
        '-- control_fkeys: No foreign keys found against table ' ||
        l_table_name || ' to ' || l_status);
    END IF;  -- no rows found
    CLOSE id_fkeys;
  END IF;  -- validated enable flag
EXCEPTION
WHEN NO_DATA_FOUND THEN -- primary key lookup failed
  p_status := 1002;
  DBMS_OUTPUT.put_line(
    '-- control_fkeys: no primary key exists for table ' ||
l_table_name);

WHEN OTHERS THEN
  p_status := SQLCODE;
  DBMS_OUTPUT.put_line('-- control_fkeys: ' || SQLERRM(p_status));
  IF (id_fkeys%ISOPEN) THEN
    CLOSE id_fkeys;
  END IF;
END control_fkeys;
/
```

*** In our current production environment, we use TOAD™ as our GUI interface to view the Oracle tables (schema and data). We added the above PL/SQL procedure to the stored procedures location in TOAD.

TESTING PL/SQL PROCEDURE
After you created your procedure, you will have to validate that the code works.
Make sure that your user-id is authorized to run such a procedure, and if not get the proper authorization from the DBA administrator.
Validation will occur through the creation of an SQL job that utilizes the CONTROL_FKEYS procedure**.**

Example of SQL:

```
variable ret_val number;
execute schema.control_fkeys('table_name',1,:ret_val);
commit;
quit;
```

** note the Control_Fkeys procedure has 3 parameters
1st Table name
2nd 1=enables constraints, 0=disable constraints (set to 1)
3rd A numeric value return value, normally 0 for success.

Submit the above SQL through SQLPLUS and review the results. The above example will enable all constraints in the named table.
The following return codes are valid:

0	=	success
1001	=	invalid enable flag(must be 0 or 1)
1002	=	no primary key exists for table

Once validated, name the SQL and place in a directory that will be used in production.
In the example above we will use '/prod/code/ enable_table.sql'. Next, proceed with the creation of the X command.

THE X COMMAND

The X command or call system command is very simple. After you have loaded your data onto the Oracle table, you will want to enable constraints. By adding the following line to your SAS code, SQLPLUS will commence. It runs a SQL script that utilizes the CONTROL_FKEYS procedure**.**

Example of X command:

```
X 'sqlplus -s userid/password @/prod/code/enable_table.sql';
```

The above X command may take some time to run, depending on how many constraints have to be enabled for that particular table.

CONCLUSION

We chose to demonstrate use of the X commands using the constraints example. Keep in mind that X commands can be used for **any** PL/SQL maintenance you need. Once the PL/SQL procedure has been developed and tested, a simple SQL job can initiate the procedure. The X command enables the programmer to have freedom. Go ahead and take your vacation. The X command will take care of the rest.

REFERENCES

Rodgers, Tony 2004. THE CONTROL_FKEYS PROCEDURE. SunTrust Bank, Atlanta, Georgia.

ACKNOWLEDGEMENTS

The authors would like to express their appreciation to the following people for their assistance in this paper:

Patrick Ryan – SunTrust Bank
Tony Rodgers – SunTrust Bank
Randall Anderson – SunTrust Bank

CONTACT INFORMATION

David Steves
SunTrust Bank
303 Peachtree Ctr. N.E.
Atlanta, GA 30303
(V) 404.827.6581
(F) 404.658.4083
david.steves@suntrust.com

Denise A. Figliozzi
SunTrust Bank
303 Peachtree Ctr. N.E.
Atlanta, GA 30303
(V) 404.230.5547
(F) 404.658.4083
denise.figliozzi@suntrust.com