Paper 323-2013

Quick and easy techniques for fast data extraction

Chris Greni, Mythili Rajamani, Deepa Sarkar, Jason Yang
Kaiser Permanente

ABSTRACT

Working with large amounts of data is a challenging, especially time consuming job. How can one reduce the amount of CPU usage to retrieve specific data? This paper tells some of the easy useful data extraction tips using SQL procedure. The following subjects are discussed in this paper:

- (1) Creating a temporary table with the Key Column in the database(both DB2, TERADATA) and extract the data from the database.
- (2) Extracting data only for specific day or specific week
- (3) Automating the date parameter for repetitive/scheduled tasks.

INTRODUCTION

Users of SAS® System software are often confronted with the challenge of retrieving specific information from very large datasets. How can the desired information be extracted effectively while reducing the amount of time required retrieving the data? For example, if your manager wants all the Tuesday's data for a year and he/she wants them in few minutes how do you do that? This paper shows how to use PROC SQL to extract the data we want especially, if the data set is large or if we do not know the "EXACT" string value and how to set up the fixed length if the length of the column (character) is set to default length.

CREATING NEW TEMP TABLE IN DB2 AND TERADATA ENVIRONMENT

Let us say a SAS® user wants to use the Primary column of the SAS® data set to extract the corresponding records from DB2 .The following SAS® PROC SQL can be used to create a temporary DB2 Table using the SAS® data set, which can be merged with the DB2 table during data extraction.

EXAMPLE 1:

Step 1:

```
PROC SQL;
     CONNECT TO DB2 (SSID = ####);
EXECUTE(DROP TABLE  LIBRARY1.TEMPDT)
           BY DB2;
           PROC DBLOAD DBMS = DB2 DATA = SAS _DATA1;
                       SSID = ####;
                       TABLE = LIBRARY1.TEMPDT;
LIMIT =150000; LOAD;
RUN;
```

Step 1 creates a DB2 table TEMPDT from the table WORK. SAS_DATA1.

Step 2:

```
PROC SQL;
     CONNECT TO DB2 (SSID = ####);
CREATE TABLE NEW_TEST AS
             SELECT * FROM CONNECTION TO DB2 (
               SELECT A.COL1,
               A.COL2, A.COL3
              FROM
                         YYY.DB2_TABLE   A,
                         LIBRARY1.TEMPDT   B
WHERE A.COL1 = B.COL1
               AND A.COL2 = B.COL2); QUIT;
```

Step 2 uses the table create in Step 1 and another DB2 table for data extraction.

Note: To create a temporary table in Teradata the following technique(Syntax) can be used.

```
%MACRO LOAD_TERADATA;
 LIBNAME USHARE TERADATA
   USER = ######
   PW = "#####"
   DB =   ####_USHARE
   TDPID = TDPN;
PROC DELETE DATA=USHARE.TEMPDT;RUN/* Delete the table if already
exists*/
  DATA USHARE.TEMPDT;
   SET SAS_DATA1;
  RUN;
%MEND;%LOAD_TERADATA;
```

Above code creates the TERADATA table TEMPDT from the table WORK. SAS_DATA1.

USING PROC SQL WITH ORACLE: USEFUL SELECT STATEMENTS

If you are extracting data from a production database that is being used by multiple users at once, you must make sure your SAS® programs are very efficient. This means we should write SQL that Oracle can understand and recognize. The following SELECT statements demonstrate usage of some useful functions within the SELECT statement to retrieve information from Oracle database in an efficient way. Using these techniques we can extract data with fewer and shorter statements than traditional SAS® code. Additionally, it uses less resources than conventional DATA and PROC steps.

EXTRACTING DATA FOR A SPECIFIC DAY OF THE WEEK

If you want to extract data for all MONDAYs for the entire year, the KEYWORD "DY" can be used in the select statement. This way we do not have to pull data for the entire year and write additional data steps to pick up only the **MONDAYs.**

EXAMPLE 2 :

```
PROC SQL;
     CONNECT TO ODBC AS REMOTE (DSN=#### UID =##### PWD =######);
     CREATE TABLEDAY_YR AS
     SELECT *

  FROM CONNECTION TO REMOTE
           (
            SELECT LOCATION,
            DATE,
             SEQUENCE_ID,
             AGE
            FROM

           SCHEMA_NAME. TABLE_NAME   A

            WHERE
             CALL_DATE >= TO_DATE('01-01-2012','MM-DD-YYYY') AND
             CALL_DATE < TO_DATE('01-01-2013','MM-DD-YYYY') AND
            TO_CHAR(CALL_DATE, 'DY') in ('MON'));
     QUIT;
```

The above PROC SQL produces the result below (records for the particular day of the 2012 in this case 'Monday').

Table 1: Data for all 'Monday's of the year 2012

	LOCATION	DATE	SEQUENCE_ID	AGE
1	OAKLAND	06FEB2012:21:32:11	1	27
2	OAKLAND	13FEB2012:21:32:22	1	79
3	OAKLAND	20FEB2012:21:33:32	2	36
4	OAKLAND	27FEB2012:21:33:35	1	17
5	OAKLAND	05MAR2012:21:32:32	1	30
6	OAKLAND	12MAR2012:21:31:09	2	2

SPECIFIC WEEK, DATA EXTRACTION

To extract the data for particular week data of the year and particular week of each month of the year from an Oracle database, adding the keywords 'WW' & 'W' in the SAS® PROC SQL returns the desired output .

EXAMPLE 3:

```
PROC SQL;
      CONNECT TO ODBC AS REMOTE(DSN = #### UID = #### PWD =######);
      CREATE TABLE WEEK_YR AS
      SELECT *

FROM CONNECTION TO REMOTE
               (
                  SELECT
                  LOCATION,
                  DATE,
                  SEQUENCE_ID,
                  AGE
                  FROM

                  SCHEMA_NAME. TABLE_NAME   A

                              WHERE
             A.DATE >= TO_DATE('01-01-2012','MM-DD-YYYY') AND
              A.DATE < TO_DATE('01-01-2013','MM-DD-YYYY') AND
             TO_CHAR(A.DATE, 'WW') = 4);
  QUIT;
```

The above PROC SQL produces the result below (records for the particular week of the 2012 in this case 4th week).

Table 2: Data for the 4th week of the year 2012.

	LOCATION	DATE	SEQUENCE_ID	AGE
1	VALLEJO	22JAN2012:00:00:14	1	91
2	VALLEJO	22JAN2012:00:01:09	1	33
3	VALLEJO	22JAN2012:00:01:13	1	0
4	VALLEJO	22JAN2012:00:01:45	1	38
5	VALLEJO	22JAN2012:00:01:53	1	30
6	VALLEJO	22JAN2012:00:01:53	1	3
7	VALLEJO	22JAN2012:00:01:53	1	2

Replacing the Selection Keyword 'WW' to 'W' in the EXAMPLE 3 will produce the following result.
(Records for the particular week of each month in 2012 in this case 4th week of each month).

Table 3: Data for the 4th week of the Feb, 2012.

	LOCATION	DATE	SEQUENCE_ID	AGE
151696	OAKLAND	24FEB2012:12:49:05	1	8
151697	VALLEJO	24FEB2012:12:49:06	1	59
151698	SAN JOSE	24FEB2012:12:49:08	1	64

Table 3a: Data for the 4th week of the Mar, 2012.

	LOCATION	DATE	SEQUENCE_ID	AGE
447026	SAN JOSE	23MAR2012:15:11:30	2	27
447027	OAKLAND	23MAR2012:15:11:30	1	4
447028	OAKLAND	23MAR2012:15:11:32	1	73
447029	SAN JOSE	23MAR2012:15:11:34	1	46

EXTRACTING DATA USING SOUNDEX FUNCTION IN SELECT STATEMENT

In Oracle the **SOUNDEX function** returns a phonetic representation, the way it sounds, of a string.
Suppose you want to extract a data with the names that are sounds alike or spelling variation from specific column then "SOUNDS - LIKE " operator is the method. To illustrate how this
Sounds – like operator works, lets extract the data for the name "Son "from the table TABLE_NAME

Let us assume the variable "NAME" in the table TABLE_NAME has the following values

SON
SUN
SUNN
DAVE

EXAMPLE 4:

```
PROC SQL;
     CONNECT TO ODBC AS REMOTE(DSN=####  UID = #### PWD =
######);
     CREATE TABLE PHNT AS
     SELECT *

  FROM CONNECTION TO REMOTE
          (SELECT *FROM

            SCHEMA_NAME.TABLE_NAME A

       WHERE
      A.DATE >= TO_DATE('01-01-2012','MM-DD-YYYY') AND
      A.DATE< TO_DATE('01-01-2013','MM-DD-YYYY') AND
      SOUNDEX (NAME) =SOUNDEX ('SON'));
     QUIT;
```

The above code will return the data for the values " SON", "SUN", "SUNN"
Note: A similar SAS® function is also available (=*), which produces the same result in SAS® environment.

AUTOMATIC DATE PARAMETER

Using macro variables which were generated by the SAS® system will reduce or eliminate the risk of assigning incorrect time range manually for the production programs. The production programs could utilize the schedule Task from windows control panel to make them run automatically with minimum maintenance efforts. Suppose you want to extract the data for every day, month, quarter etc. The following technique will be really useful. You do not need to set the Date parameter each and every time.

EXAMPLE 5:

```
%GLOBAL RDT ;

DATA NEW;
LENGTH RDT 8;

     RDT = INPUT (SYMGET ('RDT'), YYMMDD10.)  ;

IF   RDT =.
THEN RDT = TODAY ();

     YEAR = YEAR (RDT);
QTR = QTR (RDT) ;

SDT = INTNX ('QTR', RDT, -1) - 1;
EDT = INTNX ('QTR', RDT, 0);
DSN ='Y'!! PUT( INTNX( 'QTR' , RDT , -1 ) , YYQ. ) ;

     SDT_C = "'"!!PUT (SDT   , YYMMDD10.)!! "'" ;
     EDT_C = "'"!!PUT (EDT   , YYMMDD10.)!! "'" ;
     SDT_H = "'"!!PUT (SDT+1, YYMMDD10.)!!" 00:00:00'";
     EDT_H = "'"!!PUT (EDT, YYMMDD10.)!!" 00:00:00'";

CALL SYMPUTX ('SDT', SDT_C);
CALL SYMPUTX ('EDT', EDT_C);
CALL SYMPUTX ('SDT_H', SDT_H);
CALL SYMPUTX ('EDT_H', EDT_H);
CALL SYMPUTX ('DSN', DSN);

RUN;

%PUT SDT = &SDT EDT = &EDT SDT_H = &SDT_H EDT_H = &EDT_H DSN = &DSN;
```

Below SAS® program uses the macro variables SDT_H & EDT_H that are created from the above data step.

```
PROC SQL;
    CONNECT TO ODBC AS REMOTE(DSN = #### UID = #### PWD =######);
    CREATE TABLEQTR AS
     SELECT *
               FROM CONNECTION TO REMOTE
                (
                   SELECT

                         LOCATION,
                         DATE,
                         SEQUENCE_ID,
                         AGE
                         FROM

                         SCHEMA_NAME. TABLE_NAME   A

                     WHERE

              A.DATE >= TO_DATE(&SDT_H , 'yyyy-mm-dd hh24:mi:ss')AND
             A.DATE < TO_DATE(&EDT_H , 'yyyy-mm-dd hh24:mi:ss'));

QUIT;
```

CONCLUSION

This paper is intended to present some exciting features found in PROC SQL and other techniques which can be used to perform an assortment of common everyday tasks.
This paper shows that you can save considerable amounts of time by using these PROC SQL techniques and reduce the risk of assigning incorrect date/time range for the production programs.
By meandering through these examples SAS users can implement these quick and easy tips in a variety of application situations.

REFERENCES

Simon Sheppard. "Date and Time formats" Oracle Syntax
< http://ss64.com/ora/syntax-fmt.html >

ACKNOWLEDGMENTS

Special thanks to Appointment & Advice Call Center Reporting team members for their support.

CONTACT INFORMATION

Your comments and questions are valued and encouraged. Contact the authors at:

Mythili Rajamani
Kaiser Permanente
1800 Harrison St,
Oakland CA- 94588
Phone: 510-625-3252;
E -mail: Mythili.x.Rajamani@kp.org

Jason Yang
Kaiser Permanente
1800 Harrison St,
Oakland CA- 94588
Phone: 510-625-7024;
E -mail: Jason.z.Yang@kp.org

Chris Greni
Kaiser Permanente
1800 Harrison St,
Oakland CA- 94588
Phone: 510-625-6895;
E -mail: Chris.Greni@kp.org

Deepa Sarkar
Kaiser Permanente
1800 Harrison St,
Oakland CA- 94588
Phone: 510-625-6884;
E -mail: Deepa.x.Sarkar@kp.org