

# ADVANCED SQL AND DATABASE CONNECTIVITY

C. Michael Whitney, USAF Environmental Technical Application Center

---

## ABSTRACT

This paper will discuss SAS® 6.07's SQL Pass-Through Facility and how it eliminates the need for access and view descriptors when retrieving from and updating DB2® databases. Techniques described can be applied to other relational databases. Several of SAS SQL's other features will also be covered, including SQL and macro variables, summary functions, the case expression, and multi-table selects.

## INTRODUCTION

Of all the features that SAS provides us, the one that I use the most is SQL. The U.S. Air Force Environmental Technical Application Center has several hundred DB2 tables that are accessed on a daily basis. In earlier version of SAS, we were forced to manually create a new access and view descriptor for each table we wanted to use with SAS. With the release of SAS 6.07, we were freed of this slow drudgery. The tool that liberated us was the SQL's new Pass-Through Facility. With this tool, we can now communicate directly to the DB2 databases without the need for the old access and view descriptors. SQL also has the added bonuses of being able to manipulate a lot of the data in a wide variety of ways without extra SAS coding.

This tutorial assumes that the reader has at least a basic understanding of the SAS SQL language.

All of the techniques discussed here can be used on SAS datasets instead of DB2 tables. The word 'table' can be used interchangeably with 'dataset', as SQL can be used as easily on a dataset as on a DB2 table.

## THE PASS-THROUGH FACILITY

There are five components of the Pass-Through Facility. I'll briefly describe them below, and then give an example of how they are used.

- |                         |                                                                                                                  |
|-------------------------|------------------------------------------------------------------------------------------------------------------|
| CONNECT Statement       | - establishes a connection to the relational database system.                                                    |
| DISCONNECT Statement    | - cuts the link forged with the Connect statement.                                                               |
| EXECUTE Statement       | - sends SQL statements to the database.                                                                          |
| CONNECTION TO component | - allows SAS to use the database information within an SQL query.                                                |
| Return Codes            | - two macro variables &SQLXRC and &SQLXMSG that hold the return code and error message from the database system. |

Let's create a SAS dataset called SCOTTWND, which contains all year, month, day, hour, wind speed and direction variables from the DB2 database for Scott AFB surface observations, between 1971 and 1991.

```

PROC SQL NOPRINT;
CONNECT TO DB2(SSID=DSN);
CREATE TABLE SCOTTWIND AS
SELECT * FROM CONNECTION TO DB2
      (SELECT YEAR, MO, DAY, HR, WSPD, WDIR
       FROM ADB.SFC724338
       WHERE YEAR BETWEEN 1971 AND 1991);
%PUT &SQLXRC &SQLXMSG;
DISCONNECT FROM DB2;
QUIT;

```

The output would look like this:

| <u>YEAR</u> | <u>MO</u> | <u>DAY</u> | <u>HR</u> | <u>WSPD</u> | <u>WDIR</u> |
|-------------|-----------|------------|-----------|-------------|-------------|
| 1971        | 1         | 1          | 1         | 6           | 125         |
| 1971        | 1         | 1          | 2         | 8           | 120         |
| 1971        | 1         | 1          | 3         | 4           | 125         |

As you can see, it's standard SQL, with some extra lines added. Here's a breakdown of what is different:

**CONNECT TO DB2(SSID=DSN);** tells SAS to connect to the relational database system. SSID is the SubSystem ID. If you leave it blank, it will default to whatever value is in the SAS system option DB2SSID = is set.

**SELECT \* FROM CONNECTION TO DB2** gets the information from the connection we've created. We can either use the catch-all asterisk, which returns the column names from DB2 as they exist in DB2, or we can specify new variable names by specifying them with the AS command: **SELECT YEAR, MO AS MONTH, DAY, HR AS HOUR, WSPD AS WINDSPD, WDIR AS WINDDIR FROM CONNECTION TO DB2.** When using the asterisk, SAS truncates any DB2 variable name that is larger than eight characters to eight. Thus, if we had asked for ELEVATION as well, it would become ELEVATIO in our SAS dataset.

**(SELECT YEAR, MO, DAY, HR, WSPD, WDIR FROM ADB.SFC724338 WHERE YEAR BETWEEN 1971 AND 1991);** is a standard SQL select, being passed to DB2. The FROM clause specifies the DB2 table by name.

**%PUT &SQLXRC &SQLXMSG;** These two macro variables contain the return codes from the relational database. It's wise to always include this line in your code for debugging reasons.

**DISCONNECT FROM DB2;** severs the connection with the database.

Now, lets create a new DB2 table holding precipitation data. Again, it'll be for Scott AFB, IL. It'll hold year, month, day, hour, precipitation amount in tenths, and precipitation type.

```

PROC SQL NOPRINT;
CONNECT TO DB2(SSID=DSN);
EXECUTE (CREATE TABLE ADB.SCOTT_PRECIP
      (YEAR NUMERIC, MO NUMERIC, DAY NUMERIC, HR NUMERIC,
       PRCPAMT NUMERIC, PRCPPTYPE CHAR(5))) BY DB2;
%PUT &SQLXRC &SQLXMSG;
EXECUTE (COMMIT WORK) BY DB2;

EXECUTE (INSERT INTO ADB.SCOTT_PRECIP
      VALUES(1993,01,01,00, 5, 'SNOW')) BY DB2;

```

```
EXECUTE (INSERT INTO ADB.SCOTT_PRECIP
VALUES(1993,01,01,03, 20, 'SNOW')) BY DB2;
EXECUTE (INSERT INTO ADB.SCOTT_PRECIP
VALUES(1993,01,01,06, 00, 'NA')) BY DB2;
%PUT &SQLXRC &SQLXMSG;
EXECUTE (COMMIT WORK) BY DB2;

DISCONNECT FROM DB2;
QUIT;
```

The EXECUTE(...) BY DB2 command passes the SQL command to the database, telling it to create the new ADB.SCOTT\_PRECIP table, and then load it with the data. Probably not the best way to load a lot of data, but it works.

## SUMMARY FUNCTIONS

SQL can perform a lot of the same functions that PROC MEANS can. These functions include:

|                |                                                                           |
|----------------|---------------------------------------------------------------------------|
| AVG, MEAN      | - finds the average/mean of values                                        |
| COUNT, FREQ, N | - finds the number of non-missing values                                  |
| CSS            | - finds the corrected sum of squares                                      |
| CV             | - finds the coefficient of variation in percent                           |
| MAX            | - finds the largest value                                                 |
| MIN            | - finds the smallest value                                                |
| NMISS          | - finds the number of missing values                                      |
| PRT            | - finds the probability of a greater absolute value of Student's <i>t</i> |
| RANGE          | - finds the range of values                                               |
| STD            | - finds the standard deviation                                            |
| STDERR         | - finds the standard error of the mean                                    |
| SUM            | - finds the sum of the values                                             |
| T              | - finds Student's <i>t</i> value                                          |
| USS            | - finds the uncorrected sum of the squares                                |
| VAR            | - finds the variance                                                      |

These functions work against the entire column. If you wanted the total number of observations in a table, you'd use SELECT COUNT(\*). To get the maximum value, you'd use MAX(column-name). DB2 returns the value name of EXPRESSN to SAS when using the summary functions. If multiple summaries are used, you'll get EXPRESSN, EXPRESS0, EXPRESS1, etc. Use the AS command to rename them to something more useful.

We're interested in finding out what the average wind speed and direction is for Scott AFB. So we use the following code, and specify what to call the averages returned:

```
PROC SQL NOPRINT;
CONNECT TO DB2(SSID=DSN);
CREATE TABLE SCOTTAVG AS
SELECT EXPRESSN AS AVGWSPD, EXPRESS0 AS AVGWDIR
FROM CONNECTION TO DB2
(SELECT AVG(WSPD), AVG(WDIR)
FROM ADB.SFC724338);
%PUT &SQLXRC &SQLXMSG;
DISCONNECT FROM DB2;
QUIT;
```

The output would look like this having one observation with two variables:

|                |                |
|----------------|----------------|
| <u>AVGWSPD</u> | <u>AVGWDIR</u> |
| 6              | 128            |

But, what if we wanted to know the monthly average by year for these two?

```
PROC SQL NOPRINT;
CONNECT TO DB2(SSID=DSN);
CREATE TABLE SCOTTAVG AS
SELECT YEAR, MO, EXPRESSN AS AVGWSPD, EXPRESS0 AS AVGWDIR
FROM CONNECTION TO DB2
  (SELECT YEAR, MO, AVG(WSPD), AVG(WDIR)
   FROM ADB.SFC724338
   GROUP BY YEAR, MO);
%PUT &SQLXRC &SQLXMSG;
DISCONNECT FROM DB2;
QUIT;
```

Here the GROUP BY clause is used to specify that we want the averages grouped by each value in our YEAR and MO columns.

The output would look like this:

|             |           |                |                |
|-------------|-----------|----------------|----------------|
| <u>YEAR</u> | <u>MO</u> | <u>AVGWSPD</u> | <u>AVGWDIR</u> |
| 1971        | 1         | 6              | 120            |
| 1971        | 2         | 10             | 117            |
| 1971        | 3         | 12             | 119            |

## MACRO VARIABLES

Macro variables can be used anywhere within the SQL code. SQL can also place values into macro variables for you.

The following code example shows how you can make your SQL code more flexible by using macro variables within it:

```
%LET DATATBL = %STR(SCOTTWND);
%LET SLCTSTR = %STR(YEAR, MO, DAY, HR, WDIR, WSPD);
%LET TBLNAME = %STR(ADB.SFC724338);
%LET WHERESTR = %STR(WHERE YEAR BETWEEN 1971 AND 1991);

PROC SQL NOPRINT;
CONNECT TO DB2(SSID=DSN);
CREATE TABLE &DATATBL AS
SELECT * FROM CONNECTION TO DB2
  (SELECT &SLCTSTR
   FROM &TBLNAME
   &WHERESTR);
%PUT &SQLXRC &SQLXMSG;
DISCONNECT FROM DB2;
QUIT;
```

Here we can use this same piece of code over and over again, and only have to change the macro variable assigns. Note that &WHERESTR could be left blank, thus giving us all of the data in the DB2 table.

Now, lets say we want to find the minimum and maximum winds for Scott AFB, and the days they occurred. Two SQL statements will be used to do this for us. The first uses the MIN and MAX summary functions, and places them into &MINWND and &MAXWND. The second will return the year, month, day and hour of whenever the minimum and maximum winds occur.

```
PROC SQL NOPRINT;
CONNECT TO DB2(SSID=DSN);
SELECT MIN, MAX
      INTO :MINWND. :MAXWND FROM CONNECTION TO DB2
      (SELECT MIN(WSPD), MAX(WSPD)
       FROM ADB.SFC724338);
%PUT &SQLXRC &SQLXMSG;
DISCONNECT FROM DB2;
QUIT;
```

```
PROC SQL NOPRINT;
CONNECT TO DB2(SSID=DSN);
CREATE TABLE MINMAX AS
SELECT * FROM CONNECTION TO DB2
      (SELECT YEAR, MO, DAY, HR, WSPD, WDIR
       FROM ADB.SFC724338
       WHERE WSPD = &MINWND OR WSPD = &MAXWND);
%PUT &SQLXRC &SQLXMSG;
DISCONNECT FROM DB2;
QUIT;
```

The output from the first would be &MINWND holding 0, and &MAXWND holding 49. Output from the second would be all observations where the winds were either 0 or 49.

Note that in the first, you must specify what values you are getting from DB2, instead of using the catch-all asterisk. In SAS/SQL, the INTO command will place the values returned into macro variables, whose names must be preceded with a colon, i.e.. :MAXWND. Also, notice the difference in how the INTO works compared to the AS. Here we list all of the variables, followed by the INTO, and then the names of the macro variables we want the values put into.

In our second SQL code, we create a dataset called MINMAX that contains the dates, wind speeds and directions of any observations that match our min and max.

## SELECTING FROM MULTIPLE TABLES

Often you will need to get information from two or more different tables. This can easily be accomplished with SQL. In the following code, we will get the forecast wind speed and direction from ADB.TAF724338, and what actually was observed for that hour from the ADB.SFC724338 table.

```
PROC SQL NOPRINT;
CONNECT TO DB2(SSID=DSN);
CREATE TABLE SCOTT2 AS
SELECT YEAR, MO, DAY, HR, TAFWSPD, TAFWDIR, WPSD AS OBSWSPD, WDIR
      AS OBSWDIR FROM CONNECTION TO DB2
      (SELECT A.YEAR, A.MO, A.DAY, A.HR, A.TAFWSPD, A.TAFWDIR,
       B.WSPD, B.WDIR
       FROM ADB.TAF724338 AS A, ADB.SFC724338 AS B
       WHERE A.YEAR = B.YEAR
       AND A.MO = B.MO
       AND A.DAY = B.DAY
```

```

        AND A.HR = B.HR);
%PUT &SQLXRC &SQLXMSG;
DISCONNECT FROM DB2;
QUIT;

```

Output would look like this:

| <u>YEAR</u> | <u>MO</u> | <u>DAY</u> | <u>HR</u> | <u>TAFWSPD</u> | <u>TAFWDIR</u> | <u>OBSWSPD</u> | <u>OBSWDIR</u> |
|-------------|-----------|------------|-----------|----------------|----------------|----------------|----------------|
| 1971        | 1         | 1          | 0         | 10             | 120            | 8              | 120            |
| 1971        | 1         | 1          | 3         | 10             | 120            | 5              | 120            |

Once again, we're specifying the names of the variables in the first SELECT statement. Here we have to because we're getting two different WSPD and WDIR values. Our first SELECT statement will plug those returned values into the new names TAFWSPD, TAFWDIR, OBSWSPD, OBSWDIR.

In the code being passed to DB2, we use table aliases to specify the columns we are getting from each of the two DB2 tables. The line FROM ADB.TAF724338 AS A, ADB.SFC724338 AS B sets up the aliases using the AS operator. Our passed SELECT statement tells SQL that we want year, month, day, hour, wspd and wdir from the ADB.TAF724338 table, and wdir and wspd from the ADB.SFC724338 table, where the dates match up in the two. Please note, that you'll only get data back when the dates match...you won't get any data from either table when the dates DON'T match up.

A multi-table select can be across many tables, but, the more tables you query, the longer the SQL takes to run.

## THE CASE EXPRESSION

The case expression is used to select result values if certain conditions are met. It can be used to bin values. The format of the case expression is:

```

CASE <case-operand>
    WHEN when-condition THEN result-expression
    <WHEN when-condition THEN result-expression>
    <ELSE result-expression>
END

```

The WHEN-THEN clauses are used to evaluate the condition, and give the result action for that condition. The option ELSE is the catch-all for anything that doesn't meet your tests. When the case-operand is specified, it is compared to the when-condition for equality. If they are equal, then the WHEN clause is true, and the result-expression occurs. If the case-operand isn't specified after the CASE, it must be specified in the WHEN filed:

```

CASE TEST
    WHEN 1 THEN 'YES'
    WHEN 0 THEN 'NO'
    ELSE 'MAYBE'
END
or
CASE
    WHEN TEST=1 THEN 'YES'
    WHEN TEST=2 THEN 'NO'
    ELSE 'MAYBE'
END

```

Let's say we want to bin surface winds at Scott AFB into several groups; Calm, Mild, Strong, and Severe. The case expression lets us do this without an extra SAS data step:

```
PROC SQL NOPRINT;
CONNECT TO DB2(SSID=DSN);
CREATE TABLE WINDGRP AS
SELECT YEAR, MO, DAY, HR, WDIR, WSPD,
       CASE
         WHEN WSPD <2 THEN 'CALM'
         WHEN WSPD <15 THEN 'MILD'
         WHEN WSPD <55 THEN 'STRONG'
         ELSE 'SEVERE'
       END AS CATEGORY
FROM CONNECTION TO DB2
(SELECT YEAR, MO, DAY, HR, WDIR, WSPD,
 FROM ADB.SFC724338);
%PUT &SQLXRC &SQLXMSG;
DISCONNECT FROM DB2;
QUIT;
```

Note that we had to put the CASE statement in the local SELECT, and not in the DB2 SELECT. Also, this shows an example of how an expression can be renamed just like a column/variable name, using the AS command. This piece of code will give us a dataset with year, mo, day, hour, wdir, wspd and category.

Output would look like this:

| <u>YEAR</u> | <u>MO</u> | <u>DAY</u> | <u>HR</u> | <u>WSPD</u> | <u>WDIR</u> | <u>CATEGORY</u> |
|-------------|-----------|------------|-----------|-------------|-------------|-----------------|
| 1971        | 1         | 1          | 1         | 6           | 125         | MILD            |
| 1971        | 1         | 1          | 2         | 1           | 120         | CALM            |
| 1971        | 7         | 1          | 6         | 30          | 190         | STRONG          |

## REFERENCES

SAS Institute, Inc (1989), *SAS Guide to the SQL Procedure: Usage and Reference, Version 6, First Edition*, Cary, NC: SAS Institute, Inc.

SAS Institute, Inc (1991), *SAS Technical Report P-221: SAS/ACCESS Software: Changes and Enhancements, Release 6.07*, Cary, NC: SAS Institute, Inc.

SAS and SAS/ACCESS are registered trademarks of SAS Institute, Inc in the USA and other countries.

DB2 is a registered trademark of IBM in the USA and other countries.

## CONTACTING THE AUTHOR

C. Michael Whitney, SSgt, USAF  
 USAFETAC / SYS  
 Bldg. 859  
 Scott AFB, IL 62225-5116  
 (618) 256-5323  
 Internet: syscmw@etacrs1.safb.af.mil