

Using the SAS ACCESS Engine for DB2 OS/390 to Bulk Load Tables

Robert E. Maitland Jr., BlueCross BlueShield of Florida, Jacksonville, Florida

Tom Weber, SAS Institute, Cary, North Carolina

George Bischoff, Bank of America, Richmond, Virginia

ABSTRACT

The Bank of America Enterprise Data Warehouse resides in a DB2 OS/390 environment. The optimal method for joining a SAS data set with a DB2 table is to convert the SAS data set to a DB2 table. SAS version 8.2 supports bulk loading capabilities (within the SAS ACCESS engine for DB2 OS/390) by utilizing the IBM stored procedure DSNUTILS. Normally, data is inserted into a DB2 table one row at a time. Now, data can be bulk loaded into the table in a fraction of CPU time. Furthermore, this utility provides additional functionality. For example, DB2 tables can now be runstat from within SAS.

INTRODUCTION

The primary sources of data for many SAS users reside in data warehouses and data marts. Many of these data warehouses and data marts are built in relational data base management systems (rdbms) that are not SAS. SAS ACCESS provides a link for SAS users to utilize the power of the rdbms to process their data before using SAS to analyze it. With version 8, SAS has enhanced this power through the SAS ACCESS LIBNAME engines. This paper will focus on using the SAS bulk load capability to load a SAS data set into a DB2 table on OS/390 via the SAS ACCESS LIBNAME DB2 engine. First, we will provide a brief discussion of why users need to create tables in the rdbms and some of the factors they must take into consideration. Then, we will compare the methodology between the bulk load and non-bulk load technique. Next, we will provide an example using the bulk load.

USER TABLES IN AN RDBMS

Data warehouses tend to be large. Data marts are built from data warehouses to provide a higher level of performance for a line of business(s). They have fewer columns and may have business rules/processing applied before loading. The data mart can be on a platform or an rdbms structure different from the warehouse. Users can do a high percentage of their work via the data mart but not all. Therefore, they have a need to go back to the warehouse to obtain information not provided in the data mart. Furthermore, there is still a need to be able to handle data from external sources. The tables in the warehouse tend to be large versus the data sets that users need to join to them. Therefore, the optimal method of joining a SAS data set to an rdbms table is to convert the SAS table to an rdbms table.

When creating tables in an rdbms, there are some performance considerations users need to be aware of. First, the index/keys of the table(s) in the rdbms should be duplicated on the user table. They do not have to have the same name, but the same columns should be on both tables. These columns must be stored in the same format. For example, if one of the keys on the rdbms table is in small integer, then the corresponding column on the user table needs to be in small integer. Creating that column in decimal will degrade performance. Next, the user needs to create an index on the table with the columns that replicate the index/keys on the rdbms table(s). The user then needs to run the program that calculates the statistics for the user table and index. These steps are required for the rdbms to use the index(s) of the tables in the rdbms.

THE ADVANTAGES OF BULK LOAD

In version 6, SAS provided the DBLOAD procedure to load tables in DB2. With version 8, users can create tables using the LIBNAME ACCESS ENGINE DB2. Both of these methods use SQL in the background to create the tables in DB2 and load the tables with SQL inserts. That is, the tables are loaded one row at a time. (Of course users can always use pass through SQL to generate the create table and SQL insert statements themselves.) This process is slow. Furthermore, DB2 writes a line to the log for each row inserted. This can cause the log file to overflow. System performance can be degraded. Also, users have to leave SAS to run the DB2 utility that computes the statistics on the user table.

Version 6 of DB2 provides the capability of calling a utility through a stored procedure. SAS version 8.2 calls the DSNUTILS procedure to perform the bulk load. Large amounts of data can then be loaded in the user table in a fraction of CPU time. Furthermore, DSNUTILS has a great deal of functionality. Users have access to that functionality through SAS. They now have more control over the parameters. For example, they can turn the log off. More importantly, the runstats utility can be called through DSNUTILS. Users no longer have to leave SAS to call the runstats utility.

THE DSNUTILS STORED PROCEDURE

When the DSNUTILS procedure is called, SAS generates the files needed for the utility:

- SYSIN
- SYSREC
- SYSPRINT.

The control statement is generated in the SYSIN file. This file contains the instructions on how to load the table and the structure of the physical sequential file used to load the table: the SYSREC file. The SYSPRINT file stores the results of the utility. By default, the SYSPRINT file is also written to the SAS LOG.

There are a number of ways and options available when calling DSNUTILS via SAS (see Weber). For this paper, we are going to focus on the process of:

- Creating the table
- Creating the files needed to load the table.
- Creating the index.
- Loading the table.
- Doing the runstats.

The data can be found in the SASHELP library. It is the company data.

CREATING THE TABLE

The DB2 table is created using a SAS data step and specifying a LIBNAME statement with the DB2 engine. This is the part of the data step that generates the DB2 table:

```
LIBNAME MYTABLE DB2 IN =
'DATABASE.TABLESPACE';

DATA MYTABLE.TEST_TABLE (
DBTYPE=(DEPTHEAD=CHAR(15) '
    JOB1=CHAR(15) '
    LEVEL1=CHAR(16) '
    LEVEL2=CHAR(13) '
    LEVEL3=CHAR(20) '
    LEVEL4=CHAR(30) '
    LEVEL5=CHAR(30) '
    N=SMALLINT')
.
.
SET TEST_DATA;
RUN;
```

The LIBNAME statement invokes the DB2 engine. Table's created/read will have the same owner/creator

as the user id. To access different tables use the AUTHID option (see SAS OnLine Documentation).

The DB2 data base and table space that the table will be loaded in are contained in the IN option.

The column names and types for the DB2 table will match the variable names and types in SAS data set found in the SET statement (TEST_DATA). The DB2 default for character variables is VCHAR. The default for numeric values is FLOAT. Using the DBTYPE option can overwrite the default values. For example, N will be created as SMALLINT in the DB2 table.

INVOKING DSNUTILS AND CREATING THE FILES NEEDED TO LOAD THE TABLE

Now we will add the options to the data step statement that will invoke the DSNUTILS utility and create the files it needs. However, we are not ready to load the table because we have not created an index. We need to runstat the table and the index. The only way of creating the index in DB2 is through the SQL pass through. We cannot create the index until we have created the table. Parameters are available which allow us to create the files and run them later.

```
LIBNAME MYTABLE DB2 IN =
'DATABASE.TABLESPACE';

DATA MYTABLE.TEST_TABLE (
DBTYPE=(DEPTHEAD=CHAR(15) '
    JOB1=CHAR(15) '
    LEVEL1=CHAR(16) '
    LEVEL2=CHAR(13) '
    LEVEL3=CHAR(20) '
    LEVEL4=CHAR(30) '
    LEVEL5=CHAR(30) '
    N=SMALLINT')
DB2LDUTIL=YES DB2LDEXT=GENONLY
DB2LDCT1='RESUME NO
    LOG NO
    STATISTICS TABLE(ALL)
    INDEX(ALL) '
DB2IN='USERID.NAME.SYSIN'
DB2REC='USERID.NAME.SYSREC'
DB2RECSP=1000 DB2SPC1=1000 DB2SPC2=1000);
SET TEST_DATA;
RUN;
```

The DB2LDUTIL option calls the DSNUTILS procedure. The default is no.

The DB2LDEXT=GENONLY causes the files to be created but keeps the utility from being invoked. The default value is GENRUN, where the files are created and the utility is invoked.

```
The DB2LDCT1='RESUME NO
    LOG NO
    STATISTICS TABLE(ALL)
    INDEX(ALL) ';
```

statement passes these DB2 options to the SYSIN file. An alternative to RESUME NO is REPLACE; but, be careful. The REPLACE parameter causes everything in the DB2 database/tablespace to be replaced. The LOG NO turns off the log. The STATISTICS TABLE(ALL) INDEX(ALL) parameters call the runstats utility.

The DB2IN and DB2REC options are used to give names to the SYSIN and SYSREC files. If names are not given then the files are generated using the user id as the first node and system generated information for subsequent nodes. Since we need to call these files in a later step, we passed the system data set names.

The next three parameters are used to increase the file space:

- DB2RECS=1000
- DB2SPC1=1000
- DB2SPC2=1000

CREATING THE INDEX

The index is created using pass through SQL and DB2 parameters.

```
PROC SQL;
CONNECT TO DB2 (SSID=XXXX) ;
EXECUTE (
  CREATE INDEX TEST ON USERID.TEST_TABLE
  (LEVEL1,LEVEL2)
  USING STOGROUP SMSSG1
  PRIQTY 72000
  SECQTY 18000
  ERASE NO
  FREEPAGE 0
  PCTFREE 0
  BUFFERPOOL BP2
  CLOSE NO)
BY DB2;
%PUT &SQLXMSG;
DISCONNECT FROM DB2;
QUIT;
```

LOADING THE TABLE AND INVOKING RUNSTATS

Now we will use the files created in the first step to load the table and invoke the run stats utility.

```
DATA MYTABLE.TEST_TABLE (DB2LDUTIL=YES
DB2TBLXST=YES
DB2LDEXT=USERUN
DB2IN='USERID.NAME.SYSIN'
DB2REC='USERID.NAME.SYSREC' ) ;
V=0;
RUN;
```

The DB2TBLXST=YES option tells SAS that the table already exists. The default is NO. DSNUTILS loads an existing table. It does not matter when the table is created.

The DB2LDEXT is now set to USERUN. This tells SAS to invoke the load utility and use files previously created. These files are specified with the DB2IN and DB2REC parameters. The call to RUNSTATS is in the SYSIN file.

Note, there is no set statement. All the information we need to load the table is in the DATA statement. However, we need something to prime the pump. Therefore, we use the assignment statement: V=0. Since V is not a column in the table, it is ignored in the processing.

SUMMARY

Version 6 of DB2 provides the capability of calling a utility through a stored procedure. SAS version 8.2 calls the DSNUTILS procedure. DSNUTILS has a great deal of functionality. We have provided an example that utilizes some of that functionality to bulk load a SAS data set into a DB2 table and do the run stats through SAS.

REFERENCES

IBM Inc. (1999), *DB2 UDB for OS/390 V6 Utility Guide and Reference*, IBM Inc.

SAS Institute Inc. (1999), "SAS ACCESS Engine DB2", *SAS Version 8 Online Documentation*, Cary, NC: SAS Institute Inc.

Weber, Tom, *Load Utility Interface*, Unpublished White Paper, SAS Institute Inc. (2000), Cary, NC: SAS Institute Inc.

SAS is a registered trademark or trademark of the SAS Institute Inc. in the USA and other countries. ® indicates USA registration.

ACKNOWLEDGMENTS

Special thanks to Hoang Le for assistance.

CONTACT INFORMATION

Your comments and questions are valued and encouraged.

Contact the author at:

Robert E. Maitland Jr.

BlueCross BlueShield of Florida

Enterprise Business Intelligence Tools Support

4800 Deerwood Campus Parkway DCC3-2

Jacksonville, FL 32246

Phone: 904-905-1401

EMAIL: bob.maitland@bcbsfl.com

APPENDIX

```
FILENAME OUT1 'C:\SSU Paper\SASHELP_COMPANY_DATA';
```

```
DATA TEST_DATA;
```

```
  INFILE OUT1;
```

```
  INPUT
```

```
  DEPTHHEAD $Char15.
```

```
  JOB1      $Char15.
```

```
  LEVEL1    $Char16.
```

```
  LEVEL2    $Char13.
```

```
  LEVEL3    $Char20.
```

```
  LEVEL4    $Char30.
```

```
  LEVEL5    $Char30.
```

```
  N         $2.;
```

```
RUN;
```

```
LIBNAME MYTABLE DB2 IN = 'DATABASE.TABLESPACE';
```

```
DATA MYTABLE.TEST_TABLE (DBTYPE=(DEPTHHEAD='CHAR(15) '
```

```
                                JOB1='CHAR(15) '
```

```
                                LEVEL1='CHAR(16) '
```

```
                                LEVEL2='CHAR(13) '
```

```
                                LEVEL3='CHAR(20) '
```

```
                                LEVEL4='CHAR(30) '
```

```
                                LEVEL5='CHAR(30) '
```

```
                                N='SMALLINT')
```

```
  DB2LDUTIL=YES DB2LDEXT=GENONLY
```

```
  DB2LDCT1='RESUME NO
```

```
            LOG NO
```

```
            STATISTICS TABLE(ALL)
```

```
            INDEX(ALL) '
```

```
  DB2IN='USERID.NAME.SYSIN'
```

```
  DB2REC='USERID.NAME.SYSREC'
```

```
  DB2RECSP=1000 DB2SPC1=1000 DB2SPC2=1000);
```

```
SET TEST_DATA;
```

```
RUN;
```

```
PROC SQL;
```

```
  CONNECT TO DB2 (SSID=XXXX);
```

```
  EXECUTE (
```

```
    CREATE INDEX TEST ON USERID.TEST_TABLE (LEVEL1,LEVEL2)
```

```
    USING STOGROUP SMSSG1
```

```
    PRIQTY 72000
```

```
    SECQTY 18000
```

```
    ERASE NO
```

```
    FREEPAGE 0
```

```
    PCTFREE 0
```

```
    BUFFERPOOL BP2
```

```
    CLOSE NO)
```

```
  BY DB2;
```

```
  %PUT &SQLXMSG;
```

```
  DISCONNECT FROM DB2;
```

```
QUIT;
```

```
DATA MYTABLE.TEST_TABLE (DB2LDUTIL=YES DB2TLXST=YES
```

```
                        DB2LDEXT=USERUN
```

```
                        DB2IN='USERID.NAME.SYSIN'
```

```
                        DB2REC='USERID.NAME.SYSREC');
```

```
  V=0;
```

```
RUN;
```