White Paper

SAS INTEROPERABILITY WITH EMC GREENPLUM

Abstract

Interoperability (in the context of connectivity) is the ability of two different products to communicate with each other, to a certain degree. This white paper walks you through a series of examples to configure and test interoperability between SAS and an EMC® Greenplum® database. Upon reviewing this paper, readers should have a better understanding of the interoperability between the two products and how to set up SAS/ACCESS for Greenplum to help you speed up your solution deployments.

April 2011

Part Number h8240

Table of Contents

Executive summary

This white paper intends to answer frequently asked questions by developers attempting to access an EMC® Greenplum® database (GPDB) from SAS.

The intended audience includes developers who are familiar with SAS or GPDB, but not both, and want to better understand how to configure both SAS and GPDB.

Connectivity between Base SAS and GPDB is established using the SAS/ACCESS interface for Greenplum. SAS/ACCESS interfaces enable SAS solutions to read, write, and update data regardless of its native database or platform.

If you are already familiar with SAS and GPDB connectivity and you are only interested in testing the connectivity between them, go directly to Appendix A: Quick start. The appendix provides a high-level overview of the steps to configure SAS/ACCESS for Greenplum using the DataDirect ODBC driver provided in the installation package.

This paper will not discuss platform dependencies or benchmark efficiencies. Advanced data management/manipulation techniques will not be discussed. This is not an attempt to teach Base SAS, SAS PROCs, or Greenplum SQL, but rather to inform the user who wants to test connectivity between SAS and GPDB.

The SAS Data Integration tools section is a brief overview of tools available for connectivity and describes multiple ways for accessing data. The Greenplum connectivity tools section introduces such tools. Test environment explains the architecture and setup. The remaining sections explain the configuration settings for the ODBC driver and SAS programs to test the connectivity, namely the LIBNAME and Pass-Through methods.

SAS Data Integration tools

SAS Data Integration tools[1] provide capabilities for enterprise data access and processing across systems and platforms. The key components of SAS Data Integration are Enterprise Data Integration, Data Quality, and Data Access Engines. The scope of this paper is on Data Access Engines and specifically SAS/ACCESS for Greenplum[2].

SAS/ACCESS interfaces are out-of-the-box solutions. It is important to recognize that this product was written specifically to run on client/server database systems, in which a separate database engine supplies data to the local application. In this paper, all the examples use SAS as the client application and GPDB as the Relational Database Management System (RDBMS). The SAS/ACCESS Fact Sheet lists relational databases, data warehouse appliances, non-relational databases, and enterprise applications that SAS/ACCESS provides access to.

[1] SAS Products & Solutions / Data Integration page at http://www.sas.com/technologies/dw/connectivity/index.html
[2] SAS/ACCESS software page at http://www.sas.com/technologies/dw/etl/access/index.html

Access to a RDBMS requires sending database commands from the client to be executed by the server. Key features of the SAS Data Integration tools are:

- Data integration – Allows you to treat data as a resource to be viewed and used in SAS programs regardless of format
- SQL support – LIBNAME and Pass-Through are the basic means of data integration
- Bulk-loading – Moves data from SAS into third-party data stores
- Temporary table support –Creates tables that can be accessed by multiple SAS processes
- Metadata integration –Maintains RDBMS metadata within the SAS Metadata Repository
- Data integrity and security – Enables encoding of RDBMS passwords and log-in and authentication

In general it is usually easier to use SAS/ACCESS ODBC to access a RDBMS. Before you can use the product you have to set up an ODBC data source that points to the database. The user needs to install the correct driver, for example, the DataDirect ODBC driver for GPDB, and then configure the ODBC data source name (DSN). This only needs to be done once for each client workstation, after which the GPDB will be accessible. Configuration settings are explained later.

Greenplum connectivity tools

This section explains how third-party applications like SAS tools can be configured to connect to GPDB. PostgreSQL provides a number of database drivers. These drivers are not packaged with the GPDB distribution. Each driver is an independent PostgreSQL development project and must be downloaded, installed, and configured to connect to GPDB.

Greenplum provides database drivers and a C API for connecting to the Greenplum database. The following connectivity tools are provided:

- **psqlODBC** is the official PostgreSQL ODBC for a number of contributors to the PostgreSQL project[3].
- **PostgreSQL JDBC Interface** is the official PostgreSQL JDBC driver. The driver is currently maintained by a number of contributors to the PostgreSQL project[4].
- **Libpq** is the C application programmer's interface (API) to PostgreSQL (and the Greenplum Database). For more information on using libpq, see the libpq - C Library in the PostgreSQL documentation driver.

[3] Postgres ODBC Driver page at pgFoundry at http://pgfoundry.org/projects/psqlodbc

[4] PostgreSQL JDBC driver page at http://jdbc.postgresql.org

- **Progress | DataDirect** offers an industry-standard ODBC driver. In this test the DataDirect ODBC driver is used to connect SAS to Greenplum via SAS/ACCESS for Greenplum. Details are presented in the following sections.

Test environment

This section describes at a high level the architecture of the environment used for testing interoperability between the products.

The software stack for the test environment uses Red Hat Linux version 5 as the operating system running on a VMware® virtual machine. The other two components installed in this environment are SAS Foundation 9.2 and Greenplum 4.0. Figure 1 shows a high-level architectural diagram and Table 1 lists all software components used in this test.

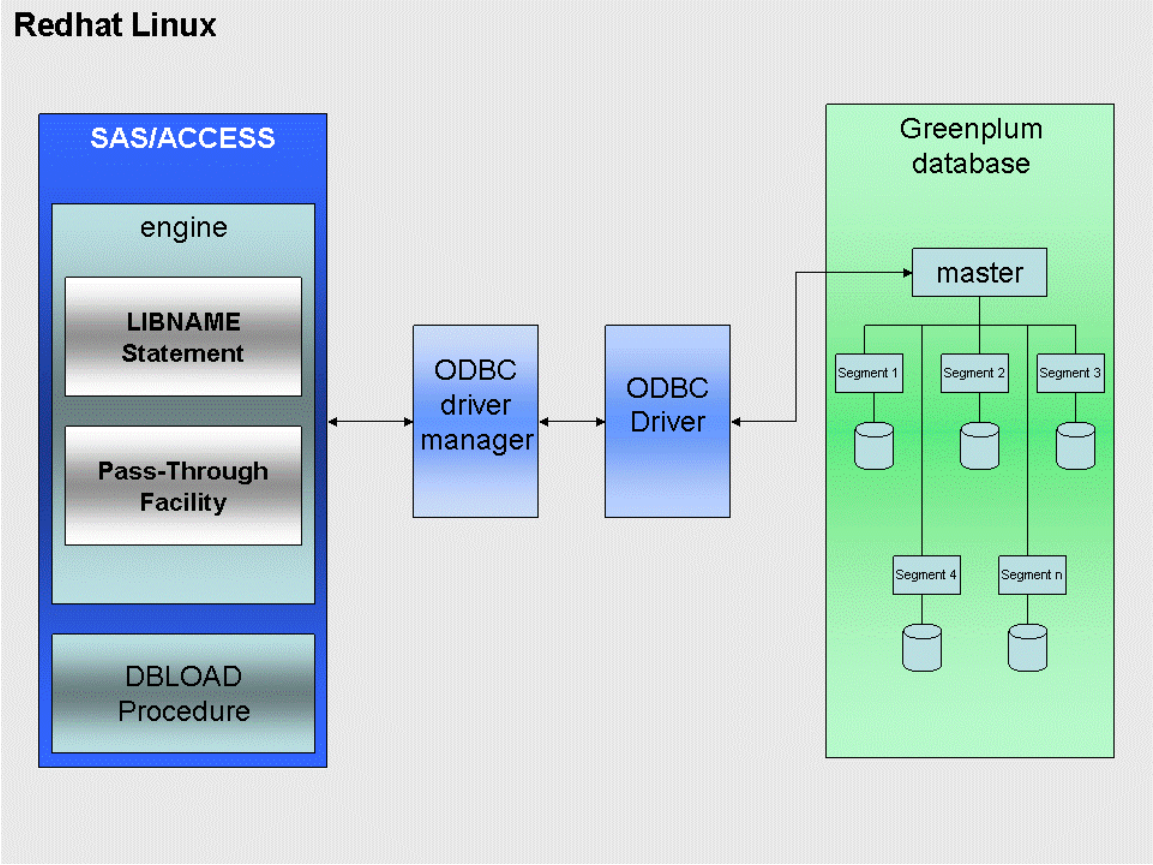

Figure 1. High-level architecture

In effect SAS/ACCESS software provides SAS-to-non-SAS database management software. SAS/ACCESS allows you to read in and modify data housed in a relational database, and then write that modified data back out to the database.

In the architectural diagram, the left-hand side shows how SAS/ACCESS for Greenplum provides three main methods for accessing the database: the DBLOAD procedure, a DBMS specific LIBNAME statement, or the SQL Pass-Through facility. The

database statements (commands) are passed to the database through the ODBC driver and the response comes back to the SAS program via the same route. On the right-hand side, the GPDB processes the Data Manipulation Language (DML) and/or Data Definition Language (DDL) commands and returns the results, when applicable, to the SAS program.

Table 1. Software components

Software	Version	Description
VMware Workstation	7.1.3 build-324285 i386	Virtual machine
SAS Foundation	9.2	SAS Foundation & SAS/ACCESS
Greenplum Database	4.0.3.0 build 5	Database management system
Progress | DataDirect	06.00.0117 (B0112, U0053)	ODBC driver for Greenplum
Red Hat Linux Server	5.3 i385	Linux operating system

Before you use any SAS/ACCESS for Greenplum features, you must install SAS Foundation, the SAS/ACCESS interface for GPDB, and configure the ODBC driver. The installation procedure is outside the scope of this paper. For details about the installation process read *Greenplum Database 4.0 Connectivity Tools for UNIX* (bundled in the connectivity tools distribution media) and the *Configuration Guide for SAS 9.2 Foundation for UNIX Environments* product documentation.

ODBC driver configuration

This section explains the key configuration settings in the "odbc.ini" file that need to be modified for the connectivity test described next. The "odbc.ini" file contains a section heading that provides a name for the driver (also known as DSN). In the example below it is "gplum". The "driver" parameter contains the path that points to the ODBC driver lib. This is the library that the driver manager will dynamically load when SQLConnect or SQLDriverConnect is called for that DSN. If this points to the wrong place, then DSN will not work.

The "Description" parameter documents the product we are using to access the GPDB. Additionally, you need to provide the following parameters: database name (Database), local hostname or IP address (HostName), Greenplum user ID (logonID), password (Password), and the port number Greenplum is listening to (PortNumber).

```
[gplum]
Driver=/GPodbc/lib/S0gplm60.so
Description=SAS ACCESS to Greenplum
Database=gpadmin
HostName=localhost.localdomain
LogonID=gpadmin
Password=gpadmin
PortNumber=5432
```

Figure 2. "odbc.ini" parameters

To verify your settings are correct, execute the "demodbc" program located in the ‹odbc home›/demo directory. The command syntax is:

```
demoodbc [-uid <username>] [-pwd <password>] [-xml] <datasource>
```

Figure 3. "demodbc" command syntax

The "demodbc" program queries the table "emp" and prints out a list of rows (see Figure 4). However, this table does not exist when you install GPDB. Use the script provided in Appendix B to create and populate the table. Alternatively, using a text editor, you can view the provided source code, "demoodbc.c", to see specifically commented code. You can customize "demoodbc.c" and build your own application by running the script "Makefile".

```
[gpadmin@localhost demo]$./demoodbc -uid gpadmin -pwd gpadmin gplum
./demoodbc DataDirect Technologies, Inc. ODBC Sample Application.
./demoodbc: will connect to data source 'gplum' as user 'gpadmin/gpadmin'.

First Name      Last Name       Hire Date    Salary      Dept
--------------- --------------- ------------ ----------- ------
George          Woltman         1982-08-07   53500       D101
Adam            Smith           1988-01-15   18000       D202
David           McClellan       1982-07-27   41500       D101
○               ○               ○            ○           ○
○               ○               ○            ○           ○
○               ○               ○            ○           ○

SQLFetch returns: SQL_NO_DATA_FOUND
```

Figure 4. "demoodbc" output

Once you successfully connect to Greenplum, execute the SAS programs in the Connectivity test section.

Connectivity test

This section presents how to establish connectivity between SAS and GPDB with a series of SAS programs to access data in Greenplum. You can invoke a SAS/ACCESS relational DBMS interface by using either a LIBNAME statement or a Pass-Through facility.

Some background is required to understand these two approaches. It is important to understand the relative similarities and differences between PROC SQL and DATA step programming. SQL has its roots in the world of relational databases. SAS was developed to manage and analyze flat files. The following table shows the equivalence of the following elements.

Table 2. Terms mapping

SAS	SQL
Data Sets	Tables
Observations	Rows
Variables	Columns
Merge	Join
Extract	Query

Using LIBNAME

The LIBNAME statement extends the SAS global LIBNAME statement to enable you to assign a "libref" to GPDB. It lets you reference a GPDB object directly in a DATA step or SAS procedure. You can associate a SAS "libref" with a database, schema, server, or group of tables and views.

A "libref" is a shortcut or a nickname for the database where the tables and views are stored. It is any SAS name (up to eight characters long) when you are assigning a new "libref". The engine name is the SAS/ACCESS engine name for the database; for the GPDB this is "greenplm". The engine name is required.

The SAS/ACCESS connection options provide connection information and control how SAS manages the timing and concurrence. For example, the basic arguments for Greenplum are user id, password, and data source name.

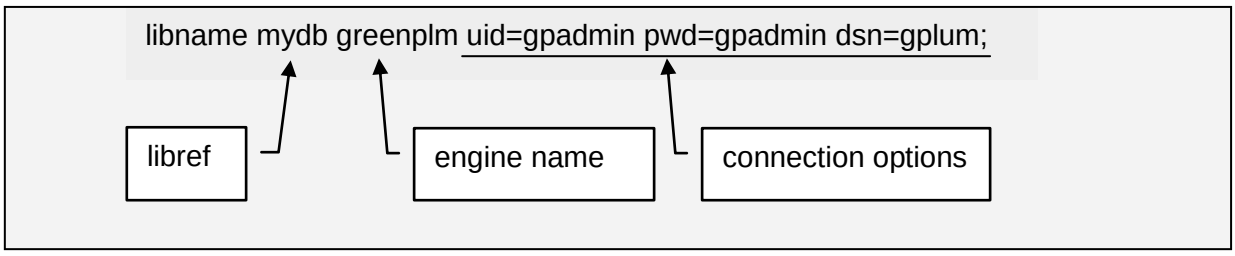

Figure 5. Libname syntax

To disassociate or clear a "libref", use a LIBNAME statement, specifying the "libref" and the CLEAR option as follows:

```
libname mydb CLEAR;
```

Figure 6. Disassociate or clear a "libref"

You can clear a single specified "libref" or all current librefs. The database engine disconnects from the database and closes any free threads or resources that are associated with that libref connection. The hello Greenplum example below shows how to create a table named "hello" with two columns: id and message. Then insert one row with id equal to "1" and a "Hello Greenplum!" message. Finally, it disconnects from the database.

Copy and paste the code in Figure 7 to a file, say "helloGPLibname.sas". To execute, open a terminal window and type "sas helloGPLibname.sas". This command will execute the program and save the output to a file named "helloGPLibname.lst". Any errors will be saved in a file named "helloGPLibname.log".

```
*read one row from the infile data proc
data new;
        input id message $3-20;
cards;
1 Hello Greenplum!;
run;

*connect to greenplum
libname mydb greenplm uid=gpadmin pwd=gpadmin dsn=gplum;

*create and populate the hello table
data mydb.hello;
        set new;
run;

* select * from table hello and print the output to the .lst file
proc print data=mydb.hello;
   title 'This is a hello message';
   run;

libname mydb CLEAR;
```

Figure 7. SAS PROC to create and populate a Greenplum table

Obviously this is simpler for users unfamiliar with SQL syntax but familiar with SAS procedures. Users with SQL experience can replace both the DATA STEP and PROC syntax shown above and embed a SQL statement using the Pass-Through facility; the next section explains how to do this. The output of the program above should look like Figure 8.

```
                                This is a hello message                 15:26
Wednesday, February 2, 2011   1

                                Obs    id    message

                                 1     1    Hello Greenplum!
libname mydb CLEAR;
```

Figure 8. Program output

Using the Pass-Through facility

The example in the previous section is repeated here using the Pass-Through facility. Pass-Through uses SAS/ACCESS to connect to the database and to send statements directly for execution. This facility is an alternative to the SAS/ACCESS LIBNAME statement. It enables you to use the SQL syntax.

The SQL procedure Pass-Through facility performs the following tasks:

1. Establish a connection with the RDBMS using a CONNECT statement and terminate the connection with the DISCONNECT statement.
2. Send non-query RDBMS-specific SQL statements to the RDBMS using the EXECUTE statement.
3. Retrieve data from the RDBMS via a SQL query with the CONNECTION TO component in the FROM clause of the SELECT statement.

Tasks No. 1 and 2 are demonstrated in the code in Figure 9. Tasks No. 1 and 3 are demonstrated in the code in Figure 10.

```
PROC SQL;
CONNECT TO greenplm AS mydb (DSN=gplum user=gpadmin
password=gpadmin);

EXECUTE (
  CREATE TABLE hello (
     id         integer,
     message    varchar(50)
  ) distributed BY (id)
) BY mydb;

EXECUTE (
  INSERT INTO hello(id, message) VALUES (1,'Hello Greenplum!')
) BY mydb;

DISCONNECT FROM mydb;
QUIT;
```

Figure 9. SAS SQL program

The following is a SAS procedure that selects the column message from the “hello” table. The SQL statement is highlighted in yellow. The “title” statement prints the “Brief message” title in the output file.

```
PROC SQL;
CONNECT TO greenplm AS mydb (DSN=gplum user=gpadmin
password=gpadmin);
%PUT &sqlxmsg;

TITLE 'Brief message';
  SELECT message
  FROM connection TO mydb (
    SELECT message FROM hello);
  %PUT &salxmsg;

DISCONNECT FROM mydb;
QUIT;
```

Figure 10. Query the hello table

The SAS %PUT statement writes the contents of the &SQLXMSG macro variable to the SAS log so that you can check for error codes and descriptive information from the Pass-Through facility. The output of the program above should look like this:

```
                                  Brief message                  15:34
Wednesday, February 2, 2011   1

                                  message
                                  ------------------
                                  Hello Greenplum!
```

Figure 11. Program output

Conclusion

The premise of this paper was to answer questions that are frequently asked by developers attempting to access a Greenplum database (GPDB) from a SAS program. The SAS system offers a variety of choices for RDBMS access, depending on the client/server platform.

It was shown that the connectivity between SAS and GPDB is established by using the SAS/ACCESS interface for Greenplum and that there are many ways to “skin a cat” using SAS software. We have demonstrated that this is certainly true when comparing PROC SQL and non-SQL Base SAS for data management techniques. While elegance versus functionality is always a consideration, given the fast pace of today's business world, demanding faster and more accurate answers, it is best that the programmer employ those techniques that are most familiar and comfortable.

Appendix A: Quick start

Prerequisites

Install the operating system, SAS Foundation 9.2, and the Greenplum database. Make sure to include the SAS/ACCESS for Greenplum interface in the options during the installation. Apply any hot fixes or patches needed. For the environment used in this paper a required hot fix was downloaded from the following URL:

http://ftp.sas.com/techsup/download/hotfix/HF2/A77.html#38046

Post-installation steps

Test your SAS installation by executing the command "sas –nodms". The output of the command should look something like this:

```
NOTE: Copyright (c) 2002-2008 by SAS Institute Inc., Cary, NC, USA.
NOTE: SAS (r) Proprietary Software 9.2 (TS2M3)
      Licensed to GREENPLUM, INC, Site 70085333.
NOTE: This session is executing on the Linux 2.6.18-128.el5 (LINUX) platform.

You are running SAS 9. Some SAS 8 files will be automatically converted
by the V9 engine; others are incompatible.  Please see
http://support.sas.com/rnd/migration/planning/platform/64bit.html

PROC MIGRATE will preserve current SAS file attributes and is
recommended for converting all your SAS libraries from any
SAS 8 release to SAS 9.  For details and examples, please see
http://support.sas.com/rnd/migration/index.html


This message is contained in the SAS news file, and is presented upon
initialization.  Edit the file "news" in the "misc/base" directory to display site-
specific news and information in the program log.
The command line option "-nonews" will prevent this display.

NOTE: SAS initialization used:
      real time           0.05 seconds
      cpu time            0.00 seconds

To exit type

 1?
```

Congratulations, your SAS installation works! Type "endsas;" at the "1?" prompt to exit.

After installation use the following steps to unpack the driver in the appropriate location.

1. CD to the ‹SASROOT›/misc/dbi directory.

2. Copy the ‹platform›gplm60.tar file to the directory where you want the Greenplum drivers to reside. The ‹platform› prefix will be unique for each operating system.
3. Change the directory to where you placed the tar file and extract the contents by issuing the following command at the UNIX prompt:

```
$ tar –xvf <platform>gplm60.tar
```

Once downloaded and installed, this directory becomes the ODBCHOME directory, which is used to set up the paths to the shared libraries as well as the “odbc.ini” file below. You must set the following environment variables:

```
export ODBCHOME=/GPodbc
export ODBCINI=$ODBCHOME/odbc.ini
export LD_LIBRARY_PATH=$LD_LIBRARY_PATH:$ODBCHOME/lib
```

Make sure the “lnxgplm60.tar” file exists in the ‹SASROOT›/misc/dbi folder. If it does not, make sure that you have downloaded and installed the hot fix mentioned above.

Updating the odbc.ini file

These are the key values in the odbc.ini file that you need to set to test the connectivity.

```
[gplum]                                  <- DSN name
Driver=/GPodbc/lib/S0gplm60.so
Description=SAS ACCESS to Greenplum
Database=gpadmin                         <- Greenplum database name
HostName=localhost.localdomain           <- Hostname or IP address
LogonID=gpadmin                          <- User Id
Password=gpadmin                         <- Password
PortNumber=5432                          <- Port Number GP is listening
```

Testing the odbc (outside of SAS)

To verify your settings are correct, execute the “demodbc” program located in the ‹ODBCHOME ›/demo directory. The command syntax is:

```
demoodbc [-uid <username>] [-pwd <password>] [-xml] <datasource>
```

The “demodbc” program queries the table “emp” and prints out a list of rows (see the figure below). However, this table does not exist when you install GPDB. Use the

script provided in Appendix B to create and populate the table. This is the output from the "demoodbc" test program.

```
[gpadmin@localhost demo]$./demoodbc -uid gpadmin -pwd gpadmin gplum
./demoodbc DataDirect Technologies, Inc. ODBC Sample Application.
./demoodbc: will connect to data source 'gplum' as user 'gpadmin/gpadmin'.

First Name      Last Name       Hire Date     Salary      Dept
--------------- --------------- ------------ -------- ------
George          Woltman         1982-08-07    53500     D101
Adam            Smith           1988-01-15    18000     D202
David           McClellan       1982-07-27    41500     D101
○               ○               ○             ○         ○
○               ○               ○             ○         ○
○               ○               ○             ○         ○

SQLFetch returns: SQL_NO_DATA_FOUND
```

Congratulations, your ODBC settings are correct! Now, let's connect from a SAS program.

Testing the SAS connectivity

Using a text editor, create a file, say "helloGPLibname.sas", and copy and paste the code below. To execute it, open a terminal window and type "sas helloGPLibname.sas". This command executes the program and saves the output to a file named "helloGPLibname.lst". Any errors will be saved in a file named "helloGPLibname.log".

```
*read one row from the infile data proc
data new;
        input id message $3-20;
cards;
1 Hello Greenplum!;
run;

*connect to greenplum
libname mydb greenplm uid=gpadmin pwd=gpadmin dsn=gplum;

*create and populate the hello table
data mydb.hello;
        set new;
run;

* select * from table hello and print the output to the .lst file
proc print data=mydb.hello;
   title 'This is a hello message';
   run;
```

The output should look like this:

```
                                  This is a hello message                  15:26
Wednesday, February 2, 2011   1

                                  Obs    id    message

                                    1     1    Hello Greenplum!
libname mydb CLEAR;
```

Congratulations, you have connected SAS to GPDB and executed a program that creates a table, populates it with one row, and performs a select * from it!

Appendix B: Creating the “emp” table

Use this script to create the table “emp” and populate it with 12 rows of fictitious employee data.

```
create table EMP (
 FIRST_NAME char(8),
 LAST_NAME          char(10),
 EMP_ID             char(6),
 HIRE_DATE          date,
 SALARY             float(2),
 DEPT               char(4),
 EXEMPT             float(2),
 INTERESTS          integer
) distributed by (EMP_ID);
insert into EMP values ('Tyler',  'Bennett',  'E10297','06/01/77',32000.00,'D101',1, NULL);
insert into EMP values ('John',   'Rappl',    'E21437','07/15/87',47000.00,'D050',1, NULL);
insert into EMP values ('George', 'Woltman',  'E00127','08/07/82',53500.00,'D101',1, NULL);
insert into EMP values ('Adam',   'Smith',    'E63535','01/15/88',18000.00,'D202',0, NULL);
insert into EMP values ('David',  'McClellan','E04242','07/27/82',41500.00,'D101',1, NULL);
insert into EMP values ('Rich',   'Holcomb',  'E01234','06/01/83',49500.00,'D202',1, NULL);
insert into EMP values ('Nathan', 'Adams',    'E41298','02/15/88',21900.00,'D050',0, NULL);
insert into EMP values ('Richard','Potter',   'E43128','04/12/86',15900.00,'D101',0, NULL);
insert into EMP values ('David',  'Motsinger','E27002','05/05/85',19250.00,'D202',0, NULL);
insert into EMP values ('Tim',    'Sampair',  'E03033','12/02/87',27000.00,'D101',1, NULL);
insert into EMP values ('Kim',    'Arlich',   'E10001','07/30/85',57000.00,'D190',1, NULL);
insert into EMP values ('Timothy','Grove',    'E16398','01/21/85',29900.00,'D190',1, NULL);
```