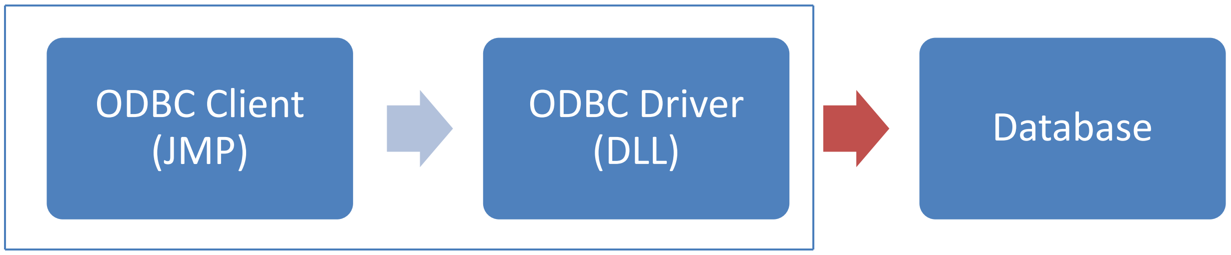
Accessing Your Database with JMP 10

JMP Discovery Conference 2012
Brian Corcoran – SAS Institute

JMP provides a variety of mechanisms for interfacing to other products and getting data into JMP. The connection to SAS, interface to R, Excel add-in and Text Import are some examples. This paper examines the ODBC (Open Database Connectivity) interface.

ODBC is a public interface that provides a general access mechanism to a wide variety of databases. Basically, the database vendor or some independent developer can write an ODBC driver. This driver will know the particulars of connecting to specific database. It will provide the standard ODBC interface to clients, interpret the calls made to the driver, and translate them for use with the database. Both the Windows and Mac operating systems support ODBC and provide ODBC Administrator applications to set up ODBC drivers. This diagram explains the connection:

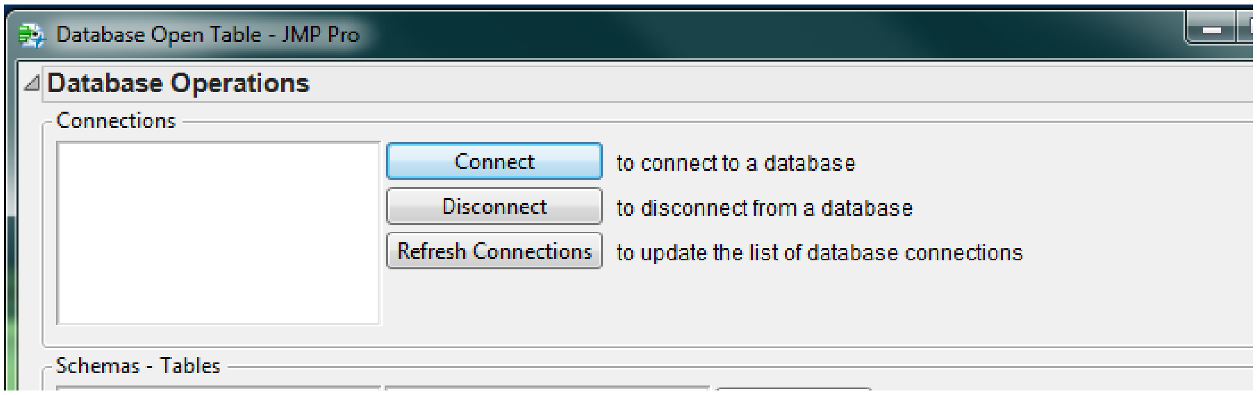


The database may be on the same machine as the JMP client, or it may be remote. The ODBC driver and the OS ODBC Administrator work together to hide the details from the client. The ODBC driver is essentially a module or DLL that is loaded into the same process space as the client. This has one very important ramification. If you are using 64-bit JMP, you must also have 64-bit ODBC drivers for the database that you intend to access. Similarly, you need 32-bit drivers if you have 32-bit JMP. You cannot mix architectures. Also, the driver must be ODBC 3.5 compliant. Most modern drivers are.

Installing an ODBC driver can take several forms. Sometimes drivers are installed with client software, like in the case of Microsoft Office. Other times, you must run a separate installer with the drivers. Once the driver is installed, you must create a DSN (Data Source Name) that contains your credentials to the database. You give this DSN a name, which will be used later to associated this group of credentials with a connection. DSN specifications differ for each database.

Once you have the DSN set up and tested, you are ready to connect to the database with JMP. JMP provides dialogs and JSL scripting support to connect to a database, issue SQL statements to the database, remove tables from the database, and save JMP tables to the database if the database permissions allow that.

To initiate a connection using the user interface, select **File -> Database -> Open Table** on the JMP menu. The first dialog that will greet you is a JMP dialog. Press the **Connect** button to bring up the ODBC Administrator dialog.


Database Open Table - JMP Pro

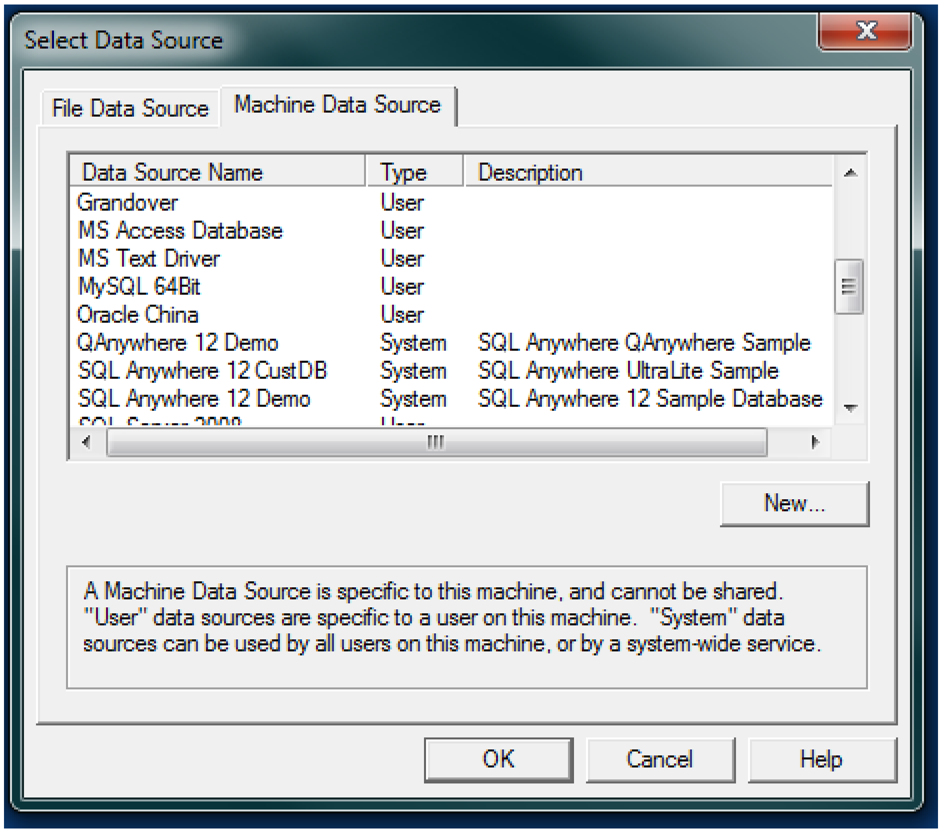
Database Operations

Connections

Connect	to connect to a database
Disconnect	to disconnect from a database
Refresh Connections	to update the list of database connections

Schemas - Tables

The ODBC Administrator is supplied by the operating system, so it will look different between Windows and the Mac, as well as between different versions of the operating system in use. JMP has no control over this dialog, and just receives the information for the DSN that the user selects. Here is the Windows version:

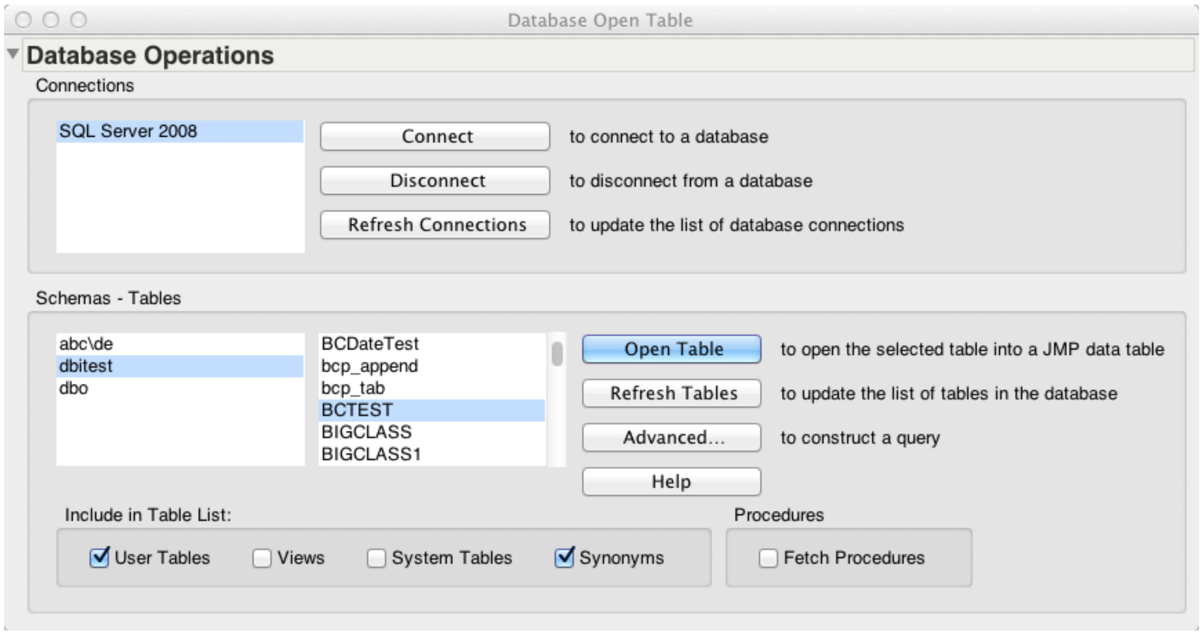

Select Data Source

File Data Source | Machine Data Source

Data Source Name	Type	Description
Grandover	User	
MS Access Database	User	
MS Text Driver	User	
MySQL 64Bit	User	
Oracle China	User	
QAnywhere 12 Demo	System	SQL Anywhere QAnywhere Sample
SQL Anywhere 12 CustDB	System	SQL Anywhere UltraLite Sample
SQL Anywhere 12 Demo	System	SQL Anywhere 12 Sample Database

A Machine Data Source is specific to this machine, and cannot be shared. "User" data sources are specific to a user on this machine. "System" data sources can be used by all users on this machine, or by a system-wide service.

Once a DSN is selected, JMP will query the database to see if it supports schemas. If it does, JMP will present a dialog with the schema information in one pane, and the tables associated with the selected schema in a second pane. If the database does not support schemas, only the table names will be shown. Checkboxes are supplied to allow the user to filter the tables shown based on whether they are System or User tables, or whether they are actually a View or Synonym. Finally, a checkbox is provided to allow you to specify that you would like a list of procedures fetched as well. Procedures are only shown in the Advanced dialog, which will be discussed later. The Open Table dialog looks about the same on Windows or the Mac. This is an example:



to connect to a database

to disconnect from a database

to update the list of database connections

to open the selected table into a JMP data table

to update the list of tables in the database

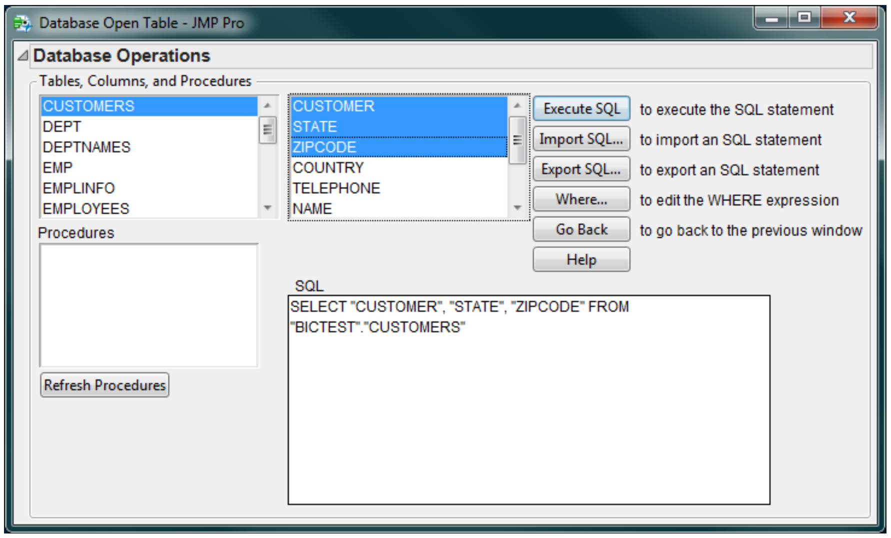
to construct a query

[✓ User Tables] Views System Tables [✓ Synonyms]

Fetch Procedures

Select the schema (if supported) and then the name of the table that you would like to open, and then press **Open Table** in the dialog. JMP will examine the variable information for the table, try to make an appropriate mapping for each variable to a JMP column, and then import the observations. JMP does not receive notifications from the database when tables are added or deleted, so if you believe the list of tables may have changed you can press the **Reset Tables** button. JMP will query the database again for the list of tables and will refresh the dialog. Once you are done accessing the database, press **Disconnect** to end the session. If you don't do this, JMP will maintain the connection until you shut down the product.

Many times you do not want the entire table imported. You may only be interested in certain variables, or the table may be really large and you may want to fetch a subset of the data. The **Advanced** button will bring up a dialog to aid in that process.



to execute the SQL statement

to import an SQL statement

to export an SQL statement

to edit the WHERE expression

to go back to the previous window

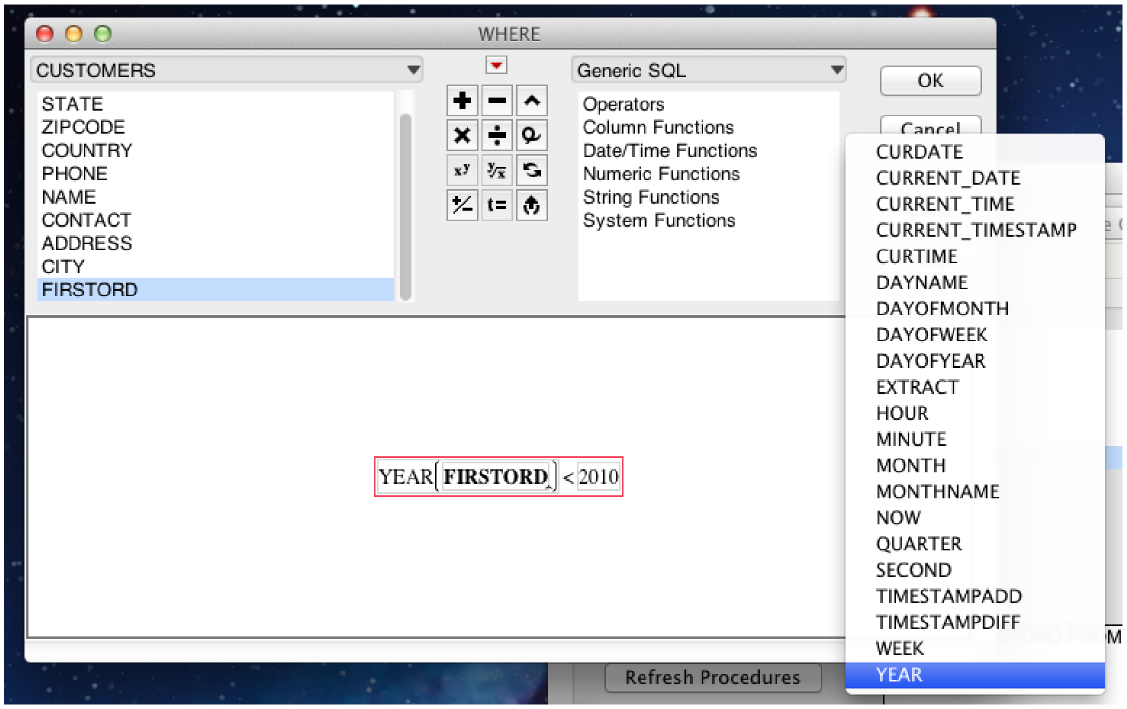
```
SELECT "CUSTOMER", "STATE", "ZIPCODE" FROM
"BICTEST"."CUSTOMERS"
```

When you first press the Advanced button, the dialog will come up with all of the variables in the table selected, and a “SELECT *” SQL statement prepared in the SQL edit box. You can select multiple variables using extended selection (Control key on Windows, Command key on Mac) and watch the SQL Select statement that is created in the SQL edit box. Once you have the desired variables selected, you can press the **Execute SQL** button to fetch the table with just those variables. If you create a complex SQL statement and want to save it for use during another session, you can use the **Export SQL** button to save the SQL in a text file for retrieval another time with the **Import SQL** button.

For users that are familiar with SQL, it is possible to just type SQL into the edit box and execute it. This allows power users to specify their own SELECT statements without having to use much of the UI.

If you have elected to query the procedures within the database, the procedures pane within the Advanced dialog will contain a list of the procedure names. Clicking on a procedure will only give the name and number of arguments that the procedure takes. You will still need to fill in the parameters and correct the syntax.

Finally, the **Where** button brings up the JMP formula editor, populated with commonly used operators for SQL statements. The operator dropdown will show Generic SQL by default, but can be set to other popular databases to get extensions specific to that database.



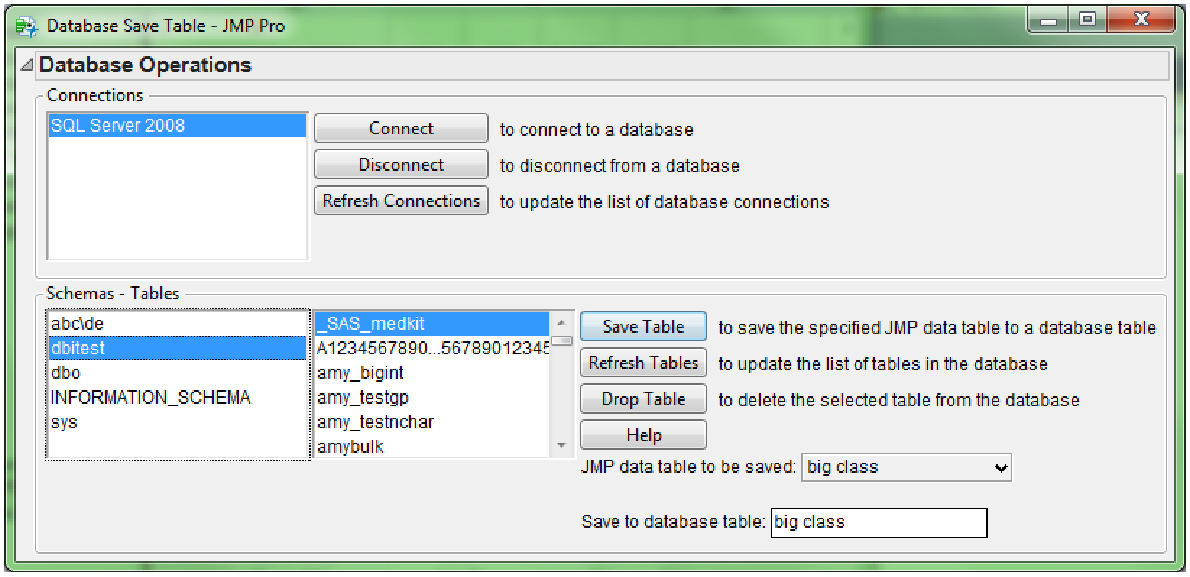
YEAR(FIRSTORD) < 2010

Once you finish the Where clause, it will be used to populate the SQL edit window back in the Advanced dialog.

Saving JMP Tables to a Database

Importing data for analysis into JMP, with the goal of producing a report of some kind, is the most typical usage scenario. However, occasionally it is desirable to save a JMP data table back to the database. JMP provides this capability, although it is up to the user and database administrator to set up the appropriate permissions to allow a SQL CREATE TABLE command to work with the database in question.

To save a JMP table back to the database, **select File->Database->Save Table** *from the table that you would like to save*. A dialog that looks similar to the Open Database dialog will come up.


Database Save Table - JMP Pro

Database Operations

Connections

SQL Server 2008

Connect to connect to a database

Disconnect to disconnect from a database

Refresh Connections to update the list of database connections

Schemas - Tables

abc\de	_SAS_medkit
dbitest	A1234567890...5678901234
dbo	amy_bigint
INFORMATION_SCHEMA	amy_testgp
sys	amy_testnchar
	amybulk

Save Table to save the specified JMP data table to a database table

Refresh Tables to update the list of tables in the database

Drop Table to delete the selected table from the database

Help

JMP data table to be saved: big class

Save to database table: big class

Enter the name that you would like to give the table in the database, and then press **Save Table**. The table list should update with your newly saved table to indicate success. The **Drop Table** button allows you to select a table to remove from the database. If for some reason you discover that the table to save is not the one that you intended, select the JMP table that you are interested in from the dropdown list.

Scripting Support

JMP supports opening database tables and saving JMP tables back to a database through the JMP scripting language. There is also some support through the Windows Automation interface. Please see the JMP Automation Reference guide for that support.

For JSL support, the Open Database command is the most common way to import a table. You must provide details about the DSN that will be used, as well as the user ID and password. For that reason, you may want to encrypt scripts that use Open Database. The parameters to Open Database provide a way to rename the table upon import and to specify if that table should be visible or invisible. An example of the syntax is:

```
dt =Open Database ("DSN=SQL Server 2008;DBQ=SQL Server;UID=test;pwd=pw1",
"SELECT * FROM dbo.prdsale", "My Product Sales Data");
```

When you import the table, as when you use the UI to import the table, the newly created JMP data table will contain a table script with the JSL syntax to recreate the table. If you do not wish for this table script to be created, you can use the “ODBC Hide Connection String” preference to suppress the script. Conversely, if you are unsure how to script the Open Database command, you can manually connect to the database and look at the resulting table script to see how the DSN is specified. The parameters for drivers often differ between Windows and the Mac, so if you are writing a script for both environments you may need to include Open Database statements for each.

Open Database opens the database, fetches and returns a reference to the new JMP table, and close the database connection all in one operation. If you would like to open a connection, execute a variety of SELECT statements and then close the connection later, the Create Database Connection, Execute SQL and Close Database Connection statements can be used. Create Database Connection returns a reusable reference handle, which is supplied to Execute SQL. An example of usage for this case is:

```
//Open a connection to a SQL Server 2008 database
dbc = Create Database Connection("DSN=SQL Server 2008;DBQ=SQL
Server;UID=test;pwd=pw1");

//Open the prdsale table and name it Product Sales
dt = Execute SQL(dbc, "SELECT * FROM dbo.prdsale", "Product Sales");

//Open the BIGCLASS table, but do not show it.
dt2 = Execute SQL(dbc, "SELECT HEIGHT,WEIGHT FROM dbitest.BIGCLASS",
Invisible, "BIG CLASS");

//Close the connection to SQL Server
Close Database Connection(dbc);
```

Finally, the Save Database command provides a way to save a JMP table back in a database. You send the Save Database message to an existing JMP table reference.

```
dt << Save Database("DSN=ORACLEDSN;DBQ=Oracle;UID=userid;pwd=pw1",
"USERTABLES.JMPCARS")
```

Save Database will typically fail if the table already exists in the database. You may need to issue a DROP TABLE statement prior to using Save Database. A replace capability is planned for a future release of JMP.

Conclusion

JMP provides both interactive and scripted mechanisms for obtaining data from most databases. Examples of databases that customers are currently accessing with JMP are Oracle®, SQL Server®, Teradata®, IBM DB2®**,** Microsoft Access®, and MySQL®.

ODBC is a mature technology, but one that continues to grow in use and importance. Microsoft's recent statement deprecating OLEDB in favor of ODBC indicates that the cross platform appeal of ODBC has generated powerful momentum for this technology.

Much of the usefulness of JMP depends on it being able to obtain and manipulate your data. The JMP development team remains committed to improving database access in each release of the product.