

Introduction

The Open Database Connectivity (ODBC) standard is a common application programming interface for accessing data files. In other words, ODBC allows you to move data back and forth easily between ODBC compliant applications. So, for example, you can easily extract data from a Microsoft Excel spreadsheet or Paradox database using SAS or SPSS without having to go through any tedious translation procedure. With ODBC, all you have to do is specify the type of file you are accessing and the data you want, and the ODBC driver will do the rest of the work.

This document is designed to show you some of the ways that you can take advantage of ODBC when using SAS and SPSS. It is assumed that the reader is familiar with SAS and/or SPSS and has access to an ODBC compliant data file. An ODBC compliant data file is a file created by a software package that conforms to the ODBC standard (e.g. Microsoft Access, Excel, Paradox, FoxPro, etc.) and has the necessary ODBC driver(s) installed. It is also assumed that the reader is going to use ODBC and SAS or SPSS on a personal computer running Windows 98/ME/NT/2000/XP.

ODBC is composed of four main components: the client application, ODBC driver manager, ODBC drivers, and the ODBC data source. The client application in this discussion will be either SAS or SPSS, and is the computer package that you will be using for your data manipulation or analysis. The ODBC driver manager is a program used to manage the links between the various applications and to configure data transfer settings. The ODBC drivers serve as the link between the client and ODBC data, and must be installed for each application. These drivers might also be composed of a network component that could be used to transfer data from a remote server. Finally, an ODBC data source is a user-generated configuration of a particular data file created using the ODBC driver manager for access by client applications.

Most ODBC applications, such as Microsoft Excel (spreadsheet program) and Microsoft Access (database program) do *not* automatically install the ODBC drivers. The ODBC drivers and driver managers are not usually included with a typical software installation and you must consult your software manual or local computer support person as to how to install ODBC for your applications.

At IUB, there are two departments in University Information Technology Services that you can consult for ODBC related problems. The [Stat/Math Center](#) (812/855-4724, statmath@iu.edu) is responsible for supporting all statistical and mathematical software packages on campus and will be able to help you with any questions relating to ODBC and statistical applications. [UITS Data Management Support](#) (dms@indiana.edu) offers a full spectrum of support services for database, spreadsheet, GIS software for personal computers, and database software on shared computers. The Data Management Services team will be able to help with any questions you have concerning ODBC and most database and spreadsheet software packages (e.g. Excel, Access, Paradox, dBase, Sybase, etc.).

Why Use ODBC?

ODBC can be used as a translation engine, facilitating the movement of data between different applications with dissimilar file structures. While most applications have some sort of import file command that performs much the same task as ODBC in this kind of situation, ODBC typically provides a more versatile and in some cases easier to use interface for transferring data between programs.

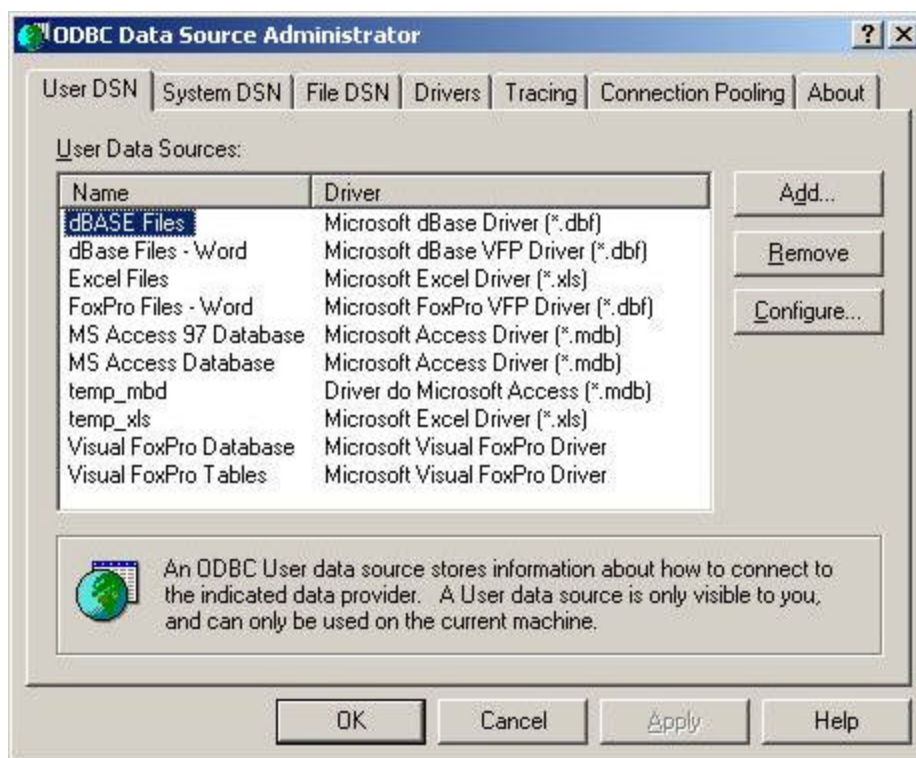
SAS and SPSS can read a variety of different file types (e.g. Lotus, Excel, dBase, etc.) without having to resort to using ODBC. With ODBC serving as the translation engine, however, you can easily access data files from any application and any version -- as long as the proper ODBC drivers are installed.

Setting Up an ODBC Data Source

Once the ODBC drivers are installed on your workstation (each software package handles the installation process

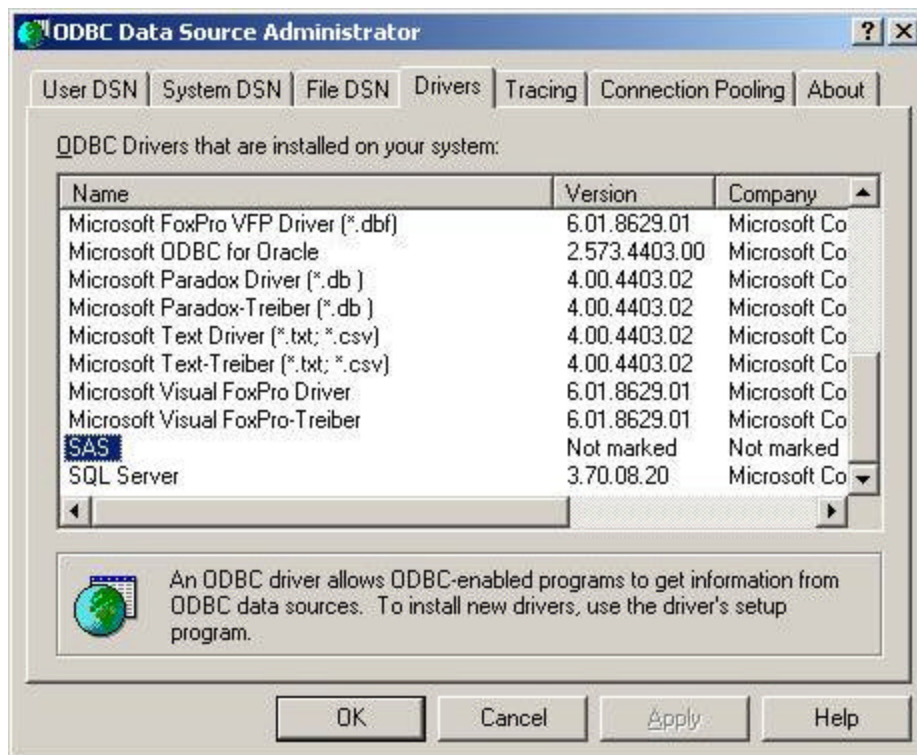
differently; consult your documentation for more information) you may have to create an ODBC data source which serves as a configuration profile for the data files. You may not have to create a data source, however, if the data files reside locally on your workstation hard disk and the client application software already created a generic data source. To check which data sources have already been created, use the ODBC driver manager. There is an icon for the ODBC Administrator located in the Control Panel under Windows 98/ME/NT/2000/XP. Other versions of Windows may or may not already have an icon for the ODBC Administrator, so a certain amount of searching may be necessary to find it. If you cannot find an icon for ODBC Administrator, check the C:\WINDOWS\SYSTEM subdirectory for the file **ODBCAD.EXE** (the 16-bit ODBC driver manager) or the C:\WINDOWS\SYSTEM32 subdirectory for **ODBCAD32.EXE** (32-bit version).

To review, modify or add ODBC data sources, run the ODBC Administrator program (also referred to as ODBC driver manager) by either double-clicking on the appropriate icon or using the **Start->Run** option in Windows. The dialogue box will look something like this:



In the text box labeled *User Data Sources*, you will find a list of all the preconfigured ODBC data sources. Again, while some of these data sources were installed along with their respective packages and no user configuration was required, some of these sources have to be added manually. If you are going to use ODBC primarily as a medium for the translation of files from one format to another, and once you install the ODBC drivers the data sources are created automatically, then you may never have to use the ODBC administrator. For example, all of the Microsoft-related data sources listed in the above dialogue box (e.g. dBASE, Excel, FoxPro, etc.) were automatically installed with Microsoft Office, so no further configuration is necessary for these data sources.

However, not all data sources are defined automatically. For example, suppose we wanted to see what ODBC data drivers are installed on a workstation. Because ODBC driver installation is independent of ODBC data source configuration, we would have to click the **Drivers** tab in the ODBC administrator to see a list of the installed drivers. The following dialogue box would then appear:



Use the scroll bar to view the entire list of drivers installed. If you compare these two examples, you can see that there are two ODBC *drivers* installed for which there are no corresponding *data sources* (SAS and SQL Server). The *Drivers* dialogue box is used only to display existing ODBC.

The next section contains a few examples of how you can create a data source. Unfortunately, it is impossible to explain in greater detail how data sources are added and configured in this discussion because the process is unique for each ODBC driver. You should consult your host application software manual for more information on how to install the ODBC driver and create a ODBC data source.

Creating an ODBC Data Source

Generally speaking, there are two types of ODBC data sources -- what one could refer to as "generic" and "file-specific" data sources. A generic data source is automatically created by some programs during the installation stage. These types of data sources, such as the ones installed by Microsoft Office, provide all of the information necessary to access a file using ODBC, except for the name and location of the file. Thus, when using a generic data source, the user must specify the drive, path, and filename interactively.

In contrast, a file-specific data source is a much more precise configuration that designates a particular file in a particular location as an ODBC data source. Creating a file-specific data source speeds up the process of accessing a file because it requires less user intervention when opening ODBC data files. A file-specific ODBC data source is usually created in situations where a single file is going to be accessed many times because it allows you to refer to the file by the assigned name rather than having to select the file each time you wish to access it.

Below are three examples of creating file-specific ODBC data sources to help illustrate how you can use this feature. The examples include creating file-specific ODBC data sources using the ODBC drivers installed by Microsoft Office and SAS.

Creating an ODBC Data Source with Excel

Microsoft Excel is a multidimensional spreadsheet program with each file containing a "workbook" composed of one

or more "worksheets." In each worksheet the data are organized by columns and rows, with the columns representing the variables, and the rows representing observations (a particular column/row combination is referred to as a "cell"). When using Excel, you have the option of using the generic data source that is supplied by Microsoft which requires you to supply the filename and location of the data file when you use ODBC to import the file into SPSS or SAS. Or you can also create a file-specific data source to simplify the task.

Regardless of whether you are using the generic ODBC data source or creating a file-specific data source, you must first assign a *name* to the data in Excel before you can import the data into SAS or SPSS using ODBC. Most casual users are not aware of the use of *names* in Excel. This feature allows you to assign a particular label to a formula or range of cells as a shortcut when referring to that object. ODBC can access only data that are labeled in this fashion. So the first step you must do to read these data using ODBC is to assign a *name* to the data you wish to import.

Imagine that you have an Excel spreadsheet called **temp.xls** located in the **c:\work** subdirectory. To prepare this file:

1. Open this file in Excel by choosing **File->Open**, typing in the location and filename (e.g. **c:\work\temp.xls**), and then click **OK**.
2. Highlight (click-and-drag) the data you wish to read using ODBC. Note: do *not* use the **Edit->Select all** shortcut to highlight the data. When you are making your selection in Excel, you should highlight only the data you wish to export, and be sure you do not include any extraneous empty columns or rows. After selecting a range of data, you must assign a name to the selection.
3. From the main menu, select **Insert->Name->Define**.
4. Type **data** in the space at the top of the dialogue box as the name of the selected data, and then click **OK**. You can define as many different selections as you like. For example you could have one name assigned to the entire file, and another to a subset of the data. In this example, however, we will only have the one selection named **data**.
5. Finally, save the spreadsheet using **File->Save** and exit Excel using **File->Exit**. You are now ready to read these data using ODBC.

At this point we could just use the generic ODBC data source to read this file, but let us automate the process a bit more by creating a file-specific data source. Do the following:

6. Launch the ODBC Administrator program.
7. When the *Data Sources* dialogue box appears, click the **Add...** button.
8. From the list of available ODBC drivers that then appears, click once on **Microsoft Excel Driver (*.xls)** and then click **Finish**.

The following dialogue box will appear:



Now you must assign a data source name. The name can be anything you want but it is always best to make the name something short and easy to remember, for you will be using this name in SAS and SPSS as a reference to this data source. For example, type **temp_xls** as the data source name. You may also include a longer description in the text box immediately below.

9. Next, click the **Select Workbook...** button to select the specific Excel file that you wish to associate with this data source name. You may use the files, directories, drives boxes and the mouse to browse through your hard disk to find the file, or simply type the drive, path and file name with extension in the *Database Name* text box in the upper left hand corner. In this case, you would type **c:\work\temp.xls** as the drive, path and filename, and then click **OK**.

10. Finally, click **OK** once more and you will return to the *Data Sources* dialogue box. If you examine the list of *User Data Sources*, you should now see a listing for the file **temp_xls**.

11. We are now finished configuring this data source, so click **OK** to exit the ODBC Administrator program.

You are now ready to use this data file with other ODBC applications.

Creating an ODBC Source with Access

Microsoft Access is a relational database program with each database file containing one or more *tables* of data. The data are organized in columns and rows, with the columns representing *fields* (variables) and the rows representing *records* (observations). With ODBC, you can read Access database or query tables. As with Microsoft Excel, you are able to use either the generic ODBC data source, or you can create a file-specific data source to simplify the task of transferring data.

The process for creating a file-specific data source is nearly identical to the one for Microsoft Excel. For example, let us say you have an Access database called **temp.mdb** located in the **C:\WORK** subdirectory, and you wished to create a file-specific data source. You would have to do the following:

1. Launch the ODBC Administrator program.
2. When the *Data Sources* dialogue box appears, click the **Add...** button.
3. From the list of available ODBC drivers that then appears, click once on **Microsoft Access Driver (*.mdb)** and then click **Finish**.

The following dialogue box will appear:



Now you must assign a data source name. The name can be anything you want but it is always best to make the name something short and easy to remember, for using this name in SAS and SPSS as a reference to this data source. For example, type **temp_mdb** as the data source name. You may also include a longer description in the text box immediately below.

4. Now you must specify the file that is going to be associated with this data source. Click the **Select...** button to assign a filename to this data source.
5. When the *Select Database* dialogue box appears, you may use your mouse to search the hard disk and subdirectories to find the file, or you may just type in the filename in the text box in the upper left hand corner. In this example, you would type **C:\WORK\TEMP.MDB**. Click **OK** to return to the dialogue box shown above.
6. Click **OK** to return to the *Data Sources* dialogue box. You will now see your new data source in the list of *User Data Sources*.
7. We are now finished configuring this data source, so click **OK** to exit the ODBC Administrator program.

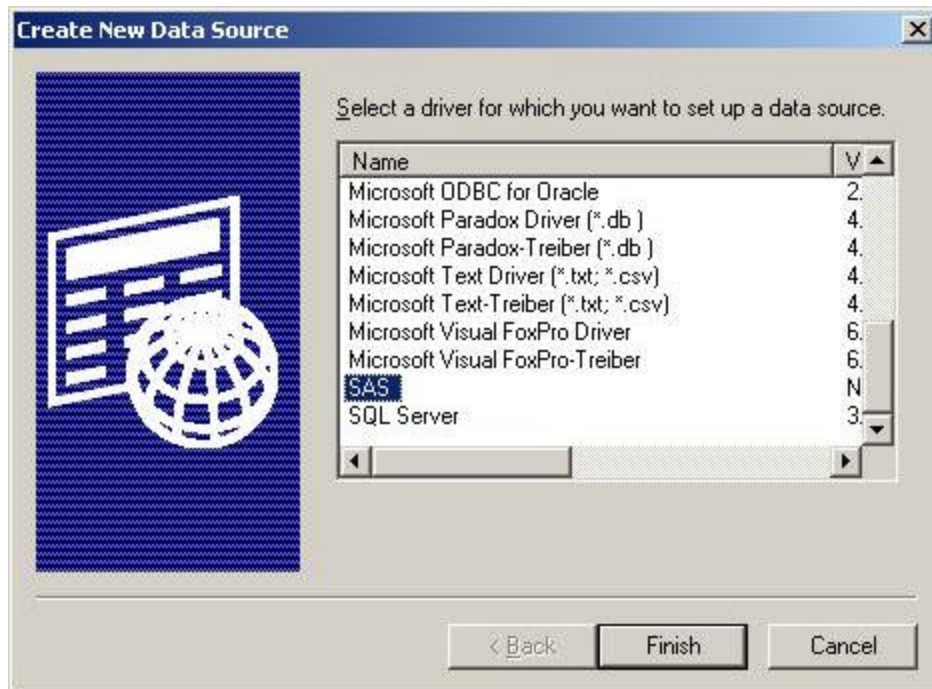
You are now ready to use this Access database with other ODBC applications.

Creating an ODBC Data Source with SAS

This example assumes that you have already installed the SAS ODBC drivers (for more information on ODBC and SAS, consult *SAS ODBC Driver: User's Guide and Programmer's Reference*) at <http://www.indiana.edu/~statmath/stat/sas/sashtml/onldoc.htm>. Imagine that you wanted to be able to read a particular SAS data set located on your hard disk into another application (e.g. SPSS). The name of the SAS data set in this example is **ODBCTEST.SAS7BDAT** and it is located in the **C:\WORK** subdirectory. To do this, you must first define an ODBC SAS data source. Unlike the Microsoft Office ODBC drivers installation, SAS does not create a generic ODBC data source, so you must create a data source for each SAS data set you wish to access. To create a SAS data source, you would have to do the following:

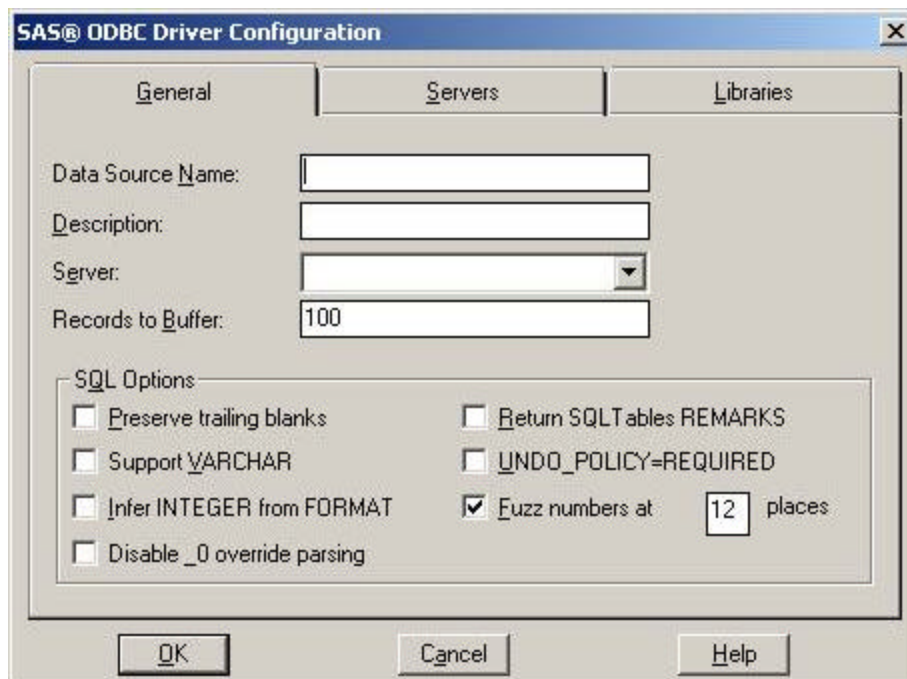
1. Launch the ODBC Administrator.
2. Add a new SAS data source by clicking the **Add** button.

You will then see something similar to the following dialogue box:



3. Select **SAS** in the *Installed ODBC Drivers* list and then click **Finish**.

You will then see the *SAS ODBC Driver Configuration* dialogue box:



4. Your next step is to assign a data source name. This can be anything but you should use something that you will be able to remember later. In this example, click once in the box labeled *Data Source Name*, and type in **ODBCTEST** as the data source name (this is the name of the SAS data set). The *Description* field is optional and it allows you to assign a full description to this data source.

5. The next step is to establish the location of the ODBC server. While SAS has the ability to serve as a remote

ODBC data server, this requires special software that is not generally available at IUB (i.e. this software does not come with the standard SAS installation and must be ordered separately), therefore the "server" you will define in this example is your own workstation. Click on the tab marked *Servers*. Next, in the box labeled *Name*, type in the name you wish to assign to your local workstation. In this example, type **local** in this box.

6. Click the **CONFIGURE** button. A dialogue box titled *Local Options* will then appear.

7. In the *Local Options* box you must specify the correct location of SAS on your hard disk, but the rest of the settings can be left at the defaults. So, for example, if SAS is located on the C: hard drive, the *SAS Path* box should read **c:\Program Files\SAS Institute\SAS\V8\sas.exe**. Once you are satisfied with the configuration, click **OK**. Make sure that Local (Single User) in the SAS Server Type box.

8. Click **<<Add<<** to store these settings. It should be noted that you will only need to assign the local workstation as a server once. Any new SAS data sources defined will automatically use the same server (i.e. your local workstation).

9. Next, you must establish a library reference, much as if you were running SAS without ODBC. Click on the *Libraries* tab and then fill in the *Name* and *Host File* boxes. In the box labeled *Name*, type in an alias for the subdirectory where your data is located. For the purposes of this example, you would type **ODBCDATA** in this box (names can be up to eight characters and can contain letters and numbers but must begin with a letter).

10. In the box labeled *Host File*, type the actual name of the subdirectory where the data is located. In our example, you would type **C:\WORK** in the *Host File* box. Click **<<Add<<** to store these settings.

11. When you are done, click **OK** to return to the *User Data Sources* box. You will now see a new data source in the list called **ODBCTEST**. This data source will allow other ODBC client applications to access the SAS data set called **ODBCTEST** in the **C:\WORK** subdirectory. To change the configuration of this data source, click once on **ODBCTEST** and then click **Configure**.

12. We are now finished configuring this data source, so click **OK** to exit the ODBC Administrator program.

These examples illustrate the various types of steps that you might have to go through to access a data set via ODBC. Because the data source configuration is dependent on the type of application used and the location of the data (i.e. local versus remote), you may have to spend quite a bit of time setting up the data sources properly. Once that step is completed, however, you will then be able to transparently move data back and forth between ODBC compliant applications.

ODBC and Statistical Software

When using ODBC and statistical software, there are a few database management concepts that you should be aware of -- tables and the Structured Query Language (SQL). Familiarity with these concepts can help you understand the structure of the database files you may be using and how they are accessed.

Most current spreadsheet and database programs organize data in a multi-tiered or three-dimensional fashion such that the data are broken down into various components, usually referred to as *tables*. For example, most modern spreadsheet programs have a *workbook* which contains individual *sheets* or *pages* each of which can contain different facets of a single data file (e.g. an accounting workbook might have a sheet for income, and another for expenditures, etc.). The majority of database management software can also organize the data in a multidimensional fashion, with each data file containing a number of different *tables*, each of which contains different elements of the entire database. So, if you had a sales database, for example, the data file might contain one table for customers' addresses, another for prices and products, another for tracking shipping, etc. So, instead of a data file containing only a single data set, it is possible for a single database to contain many data sets, or many different permutations of the same data

set. For example, a single data set could contain data along with various different views and query tables that are just variants of the original data. It is therefore vital that you know the exact structure of a data file before you attempt to read the data into another program. Each table in a database usually has an assigned name, and you must know the name of the table you wish to read using ODBC to be able to transfer the data properly.

SQL, a standardized database language designed to select data from one or more database tables, is another important database management concept. SQL is used by both SPSS and SAS to specify the range of variables and/or observations that you wish to read from an ODBC data source. This can save time in reforming a data file in SAS or SPSS by reading in only those variables and/or observations that you are interested in, saving you from having to transform the data after you have read the data. Both SPSS and SAS use standard SQL statements to read data from an ODBC source, by using the *SELECT* command to specify the variables you wish to access, and the *WHERE* command to specify the observations.

The discussions of SAS and SPSS below include examples of reading database files with multiple tables and SQL to familiarize you with these database management concepts.

SAS and ODBC

SAS uses the SAS/ACCESS module and the PROC SQL procedure to access external ODBC data sources. As with most SAS procedures, there is no point-and-click interface available and all of the steps will have to be accomplished using the SAS command language. Using PROC SQL, you can access any ODBC data source in SAS. SAS/ACCESS can be used without PROC SQL to access many popular PC-file formats (e.g. Lotus, dBASE, Excel, etc.) but it is considerably more difficult to configure and use (see *SAS/ACCESS Interface to PC File Formats* for more information). Remember, if you are importing data from a spreadsheet, you *must* first assign a name to the data you wish to import, as discussed in a previous section.

Imagine, for example, that we followed the example in the section "Creating an Excel Data Source" such that we have an Excel file called **temp.xls** in the **C:\WORK** subdirectory. You have already assigned the name **data** to a selection of data and created an ODBC data source called **temp_xls** for this file. To read this file, you would do the following:

1. Launch SAS.
2. In the *SAS Program Editor*, type the following commands:

```
PROC SQL;

CONNECT TO ODBC(DSN='temp_xls');

CREATE TABLE temp_sas AS

SELECT * FROM CONNECTION TO ODBC(SELECT * FROM data);

QUIT;
```

3. Highlight these commands and from the main menu select **Run->Submit**.

In this example, we use a file-specific ODBC data source (**temp_xls**) to read the data. You may also use a generic ODBC data source to read the same file by doing the following:

1. In the SAS Program Editor, type the following commands:

```
PROC SQL;
```

```
CONNECT TO ODBC(PROMPT);

CREATE TABLE temp_sas AS

SELECT * FROM CONNECTION TO ODBC(SELECT * FROM data);

QUIT;
```

2. Highlight these commands and from the main menu select **Run->Submit**.

3. An ODBC dialogue box will appear and prompt you for the type of file you wish to access. From the list, select **Excel Files** and then click **OK**.

4. A final dialogue box will then appear, asking for the name and location of the file you wish to retrieve. You can either use the mouse to choose the appropriate drive and subdirectory, or just type the name and location of the file in the top left hand corner of the dialogue box. In this example, you would type **C:\WORK\TEMP.XLS** and then click **OK**.

The **PROC SQL** procedure is generally used to issue SQL commands to SAS data sets but if you have the SAS/ACCESS module installed, you can use this procedure to issue queries to other types of data files including ODBC data sources. By default, an SQL query accesses a remote data file and then responds with a list of the results of your query. In these examples, we asked SAS to select all of the variables (**SELECT ***) from the ODBC data source **temp.xls**. The reason there are two **SELECT *** statements in this example is because the first describes what **PROC SQL** will select, and the second describes what the Excel ODBC driver will select. While SAS is fully SQL-compliant, many ODBC drivers can only understand rudimentary SQL commands (i.e. **SELECT** and **WHERE**), so both commands must be used to synchronize **PROC SQL** with the ODBC driver.

If you want to read the results of a **PROC SQL** into a SAS data set for analysis, you must also use a **CREATE TABLE** subcommand as above, followed by the name of the data set you wish to create. In this example, we access the Excel file called **temp.xls** and then select all of the variables and cases in that spreadsheet and store them in a SAS temporary data set called **temp_sas**.

Now let us try a more complicated example of using **PROC SQL** to read an ODBC data source. This time, imagine that you are reading a Microsoft Access database file. The file is called **temp.mdb**, it is located in the **C:\WORK** subdirectory, and you have already created an ODBC data source for the file called **temp_mdb**. The file has two tables of data (table1 and table2), each with five variables (table1 contains ID, V1, V2, V3, V4; table2 contains ID, V5, V6, V7, V8), where ID is a primary index variable (i.e. it is used to order the data), and there are 10 observations. The ID variable tracks the case number (ranges from 1 to 10) and the variables V1-V8 are numeric data. Furthermore, you want extract the last five cases from both tables, and variables V1 and V2 from Table1, and V7 and V8 from Table2. So the final data set would be composed of five cases with four variables: V1, V2, V7, and V8. The syntax you would use to complete this task would be:

```
PROC SQL;

CONNECT TO ODBC(DSN=temp_mdb);

CREATE TABLE temp_sas AS

SELECT * FROM CONNECTION TO ODBC(SELECT V1, V2, V7, V8

FROM TABLE1, TABLE2 WHERE TABLE1.ID=TABLE2.ID AND TABLE.ID > 5);

QUIT;
```

This example creates a new SAS data set called **temp_sas** using the CREATE TABLE subcommand. As you can see, the SELECT statement issued to the Access ODBC driver is much more complicated than in our previous examples. First, we use the SELECT command to list the variables we are interested in (V1, V2, V7, V8). Next, we use the FROM command to list the tables from which these variables are to be drawn (Table1 and Table2). A WHERE command is used to specify which records/observations we are interested in. In this example, the database file is indexed so we must specify that the index shared by each table is the same. Each variable/field can be referred to by including the appropriate table so TABLE1.ID=TABLE2.ID can be understood to mean that the ID variable in Table1 is set to be equal to the ID variable in Table2. Finally, an additional condition is set using the AND condition to include only those cases where ID is greater than five (i.e. the last five cases).

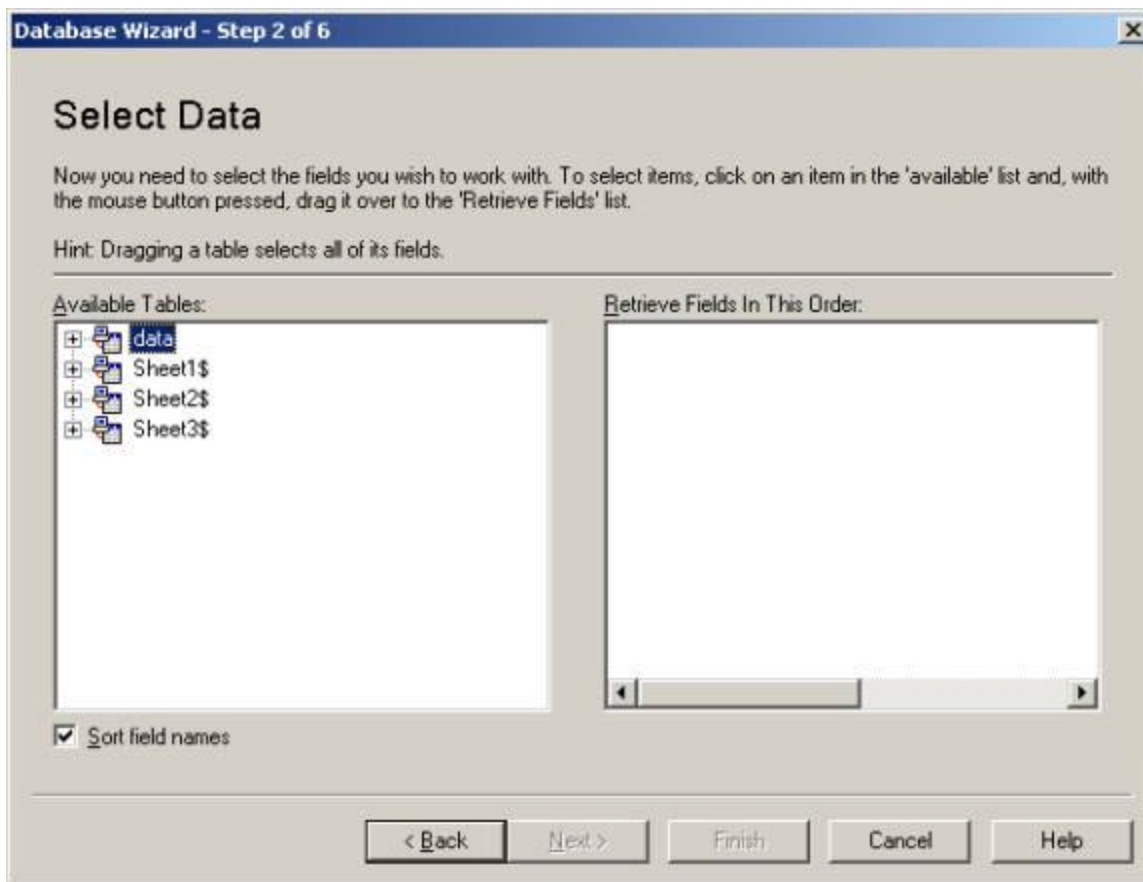
While this example is much more complicated than the most common uses of SAS and ODBC, it does demonstrate how PROC SQL can be used. Explaining all of the command choices available is beyond the scope of this discussion. For more information, consult *SAS SQL Query Window User's Guide*, and *SAS/ACCESS Software for PC File Formats*.

SPSS and ODBC

SPSS has a very easy to use point-and-click interface for accessing ODBC data sources. Remember, however, that unlike SAS, SPSS can read only ODBC data sources and cannot serve as an ODBC data source. This means you can use ODBC to read data into SPSS, but you cannot use ODBC to read data from SPSS files. Also unlike SAS, SPSS has the ability to *easily* read many different types of files, and more often than not this would meet the needs of most users. If, however, you wished to access an ODBC data source, SPSS can do so.

Once again, imagine you have an Excel file called **temp.xls** in the C:\WORK subdirectory which has a named selection called **data** and you have created an ODBC data source for this file called **temp_xls**, as in the example discussed in "Creating an Excel ODBC Data Source." To read this file into SPSS, you would do the following:

1. Launch SPSS.
2. From the main menu, select **Database ->New Query**. The *Database Wizard* dialogue box will appear.
3. In the *Database Wizard* dialogue box, select the **temp_xls** data source and then click **OK**.
4. You see the *Select Data* dialogue box, shown below:



5. In the box labeled *Available Tables*, SPSS will list all of the named selections in the spreadsheet. In this example, we have assigned a name to one selection in Excel, the one we called **data**. If there were more than one named selection, each one would be listed in the *Available Tables* box. To select a table, click on the table you wish to access and drag it to the right-hand box labeled *Retrieve Field In This Order*. In this example, you would click on the table **data** and drag the table to the right-hand box before releasing the left mouse button.

6. The right hand box will then display a listing of all of the variables/fields contained in the table you selected. You can deselect variables by **double-clicking** on each variable name. By default, the ODBC driver will assign each column a name such as **F1, F2, F3...** if there were no variables names assigned to the original data in Excel. These field names correspond to the original column order (i.e. column A equals F1, column B is F2, etc.). Click **NEXT**. The Limit Retrieved Cases dialogue box will then appear

7. The Limit Retrieved Cases dialogue box can be used to create a filter for the data, so that certain cases could be dropped during the transfer process. This function behaves much like the **SELECT IF** command in SPSS. In this example, we want to read all of the observations, so we do not have to use this option, so at this point you would simply click **FINISH** to import the data into SPSS.

SPSS will then show the results of this command. Having described a simple example, we can now explore some of the other options available. Below is a slightly more sophisticated example where a filter is applied to this process.

Imagine that you have a Microsoft Access database file called **temp.mdb**. The file is located in the **C:\WORK** subdirectory and you have already created an ODBC data source for the file called **temp_mdb**, as in the example in the section "Creating an Access ODBC Data Source." The file has two tables of data (Table1 and Table2), each with five variables (table1 contains ID, V1, V2, V3, V4; table2 contains ID, V5, V6, V7, V8), where ID is a primary index variable (i.e. it is used to order the data), and there are ten observations. The ID variable tracks the case number (ranges from 1 to 10) and the variables V1-V8 are numeric data. Furthermore, you want to extract only the last five cases for variables V7 and V8 from Table2. So the final data set would be composed of five cases with two variables.

You would need to do the following:

1. From the main menu, select **New Query...** The *Database Wizard* dialogue box will appear.
2. In the *Database Wizard* dialogue box, select the **temp_mdb** data source by clicking on the name once, and then click **NEXT**. The *Select Data* dialogue box will appear.
3. In the *Select Data* dialogue box, select the table you wish to draw the data from in the box in the upper left hand corner. In this example, you want to see a list of all of the variables/fields available in Table2. Click on the +next to the name of Table2 to see a listing of the tables.
4. Next, select the fields **V7** and **V8**. You would use the scroll bar to find these fields/variables and then select them by **double-clicking** on each variable.
5. Now you need to set the **WHERE** condition. Click on the **NEXT** button. The *Limit Retrieved Cases* dialogue box will appear.
6. In the *Limit Retrieved Cases* dialogue box, click the **Select where case satisfies condition** radio button in the top middle part of the dialogue box.
7. Double click **ID** in the **Fields** box. **Table 2:ID** appears in the **Expression 1** column. Click the cell right below the **Relation** column and type **>**. Type **5** in the cell right below the **Expression 2** column.
8. Click the **FINISH** button when done setting the condition to import the data.

The biggest advantage of using SPSS to read data by using the windows interface is that it provides you with an easy to use interface. However, you can also use the **GET CAPTURE** command to perform the same tasks described above. The following discussion demonstrates how you can use the command to replicate the results from the previous interactive examples, as well as shows how you could implement more sophisticated queries. For more information on the **GET CAPTURE** command, consult the *SPSS Base 7.0 Syntax Reference Guide*.

Assume that you have an Excel spreadsheet file with an assigned ODBC data source called **temp_xls** with a named selection called **data**, as used in the first example in this section and in the section "Creating an Excel ODBC Data Source." You could read this file into SPSS interactively as described above, or you could open a Syntax window in SPSS and use the **GET CAPTURE** command instead:

1. From the main menu, select **File->New->Syntax**.
2. In the Syntax window, type the following commands:

```
GET CAPTURE ODBC

/CONNECT='DSN=temp_xls'

/Select * FROM `data`.

EXECUTE.
```

3. Highlight these commands by either using the mouse to click-and-drag over the text or choose **Edit->Select All** from the main menu.
4. Click the **RUN** button (a right-pointing triangle on the button bar) or choose **Run->Selection** from the main menu.

The syntax for this example is similar to the SAS examples previously discussed. Both SAS and SPSS use SQL commands to query external ODBC data sources. In this example, we use the **SELECT *** command to select all of the variables from the table called **data**.

We could also use the **WHERE** command to set a condition to determine which observations are read into SPSS, just as we did in the second example above when we read the last five cases from a Microsoft Access database with two tables. Again, assume that you had an Access database with two Tables (Table1 and Table2), each table containing four variables (V1-V4 in Table1, V5-V8 in Table2), and each table shares an index variable called ID. And just as in the interactive example above, suppose you want to read the last five cases for V7 and V8 into SPSS. To do this using the GET CAPTURE command:

1. From the main menu, select **File->New->Syntax**.
2. In the Syntax window, type the following commands:

```
GET CAPTURE ODBC

/CONNECT= 'DSN=temp_mdb'

/Select V7, V8 FROM Table2

WHERE ID > 5.

EXECUTE.
```

3. Highlight these commands by either using the mouse to click-and-drag over the text or choose **Edit->Select All** from the main menu.
4. Click the **RUN** button (a right-pointing triangle on the button bar) or choose **Run->Selection** from the main menu.

Now let us consider a more complicated variation of this example, taking advantage of the GET CAPTURE command's ability to read multiple tables. Assume that we are using the same Microsoft Access database above, but suppose you want to read two variables from Table1, V1 and V2, and two variables from Table2, V5 and V6. Furthermore, you are only interested in the first five cases for each table. This time you would use the following steps:

1. From the main menu, select **File->New->Syntax**.
2. In the Syntax window, type the following commands:

```
GET CAPTURE ODBC

/CONNECT= 'DSN=temp_mdb'

/SELECT Table1.V1, Table1.V2, Table2.V5, Table2.V6 FROM

Table1, Table2 WHERE Table1.ID=Table2.ID AND

Table1.ID < 5.

EXECUTE.
```

3. Highlight these commands by either using the mouse to click-and-drag over the text or choose **Edit->Select All**

from the main menu.

4. Click the **RUN** button (a right-pointing triangle on the button bar) or choose **Run->Selection** from the main menu.

The major difference between this and the previous GET CAPTURE examples is the use of the **SELECT** and **WHERE** SQL subcommands. Also, if you look at these subcommands, you will see that we refer to each variable in the multiple-table Access database by a two-level name composed of the table name and the variable name, so Table1.V1, for example, refers to the V1 in Table1. You also have to list all of the tables from which you are drawing data with the **FROM** parameter. Finally, the **WHERE** subcommand is used to synchronize the index/sort variable (ID) in each table and the **AND** conditional statement is used to specify which cases you wish to read. This relatively simple example of how the GET CAPTURE command can be used to read multidimensional data files demonstrates some of the many options you have when reading ODBC data sources into SPSS.

Conclusion

ODBC adds another means of bringing data into SAS and SPSS. While not easily mastered, this method can provide you with a great level of flexibility and versatility if you are frequently moving data between database or spreadsheet applications and SAS or SPSS.

ODBC is by no means the exclusive way to import data into these applications but it does provide you with another possible method of accessing files that may not be easily transferred otherwise. In fact in many cases, the standard built-in translation routines that are common in statistical software such as SPSS and SAS are faster and easier to use. But there may well be situations where ODBC can provide you with an added level of versatility and performance.

Permission to use this document is granted so long as the author is acknowledged and notified.

Please send comments and suggestions to: statmath@indiana.edu

Copyright 1995-2003, [Indiana University](http://www.indiana.edu).