

9.1 Introduction:

The great thing about ACCESS is that it provides powerful and easy-to-use tools for organizing data, creating forms, producing reports, and automating tasks. These desktop tools are a significant improvement over the arcane command-line interfaces that ship with many industrial-strength database systems.

The not-so-great thing about ACCESS is that the database engine itself it is not designed to support high transaction volumes or a large number of simultaneous users. For example, one simply could not run an airline reservation system on top of an ACCESS database.

Fortunately, ACCESS can play the role of the “client” when connecting to an industrial-strength **client/server database**. The implication is that you can continue to work within ACCESS without having to store multiple copies of your organization’s data in multiple independent desktop database systems. Instead, many desktop systems can link to a single source of data that is stored and administered centrally.



You had a glimpse of how this might work in [Section 8.3.3](#) when you created a link

to a dBASE IV file containing payroll data. However, dBASE IV is definitely not a client/server database and the file we linked to was on the same machine as the ACCESS database. Client/server databases are designed to be used by multiple simultaneous users over network connections.

In order to use ACCESS to connect to a database server over a network, you need something called middleware. In this demonstration, we are going to use **Open Database Connectivity (ODBC)** middleware to create a link to data on a different computer using a different operating system and running a different DBMS.

9.1.1 Doing versus demonstration

What I used to do with my students is provide them with the Internet Protocol (IP) address of a SQL SERVER database server, give them a valid user name and password for the server, and make them create an ODBC connection to the SQL SERVER database over the Internet. Although this was certainly a cool exercise when it worked, the more typical outcome was conflicting ODBC driver versions, flaky networks, and frustration. And all these



problems occurred in (what was supposed to be) a standardized computing lab environment.



Client/server computing relies on reliable high-speed network connections between the clients and the server. Without robust networking infrastructure, and compatible middleware, client/server is more trouble than it is worth.

My new approach is to simply demonstrate the process of setting up an ODBC connection to a client/server database. Although a passive demonstration is no substitute for hands-on experience, it is simply not practical to let everyone who works through these tutorials access my database server.

Of course, it is possible to set up your own client/server database in order to work through this lesson.¹ Just be warned that setting up a client/server database server is tricky and is probably not worth the effort and angst at this early stage.

¹ ACCESS 2000 includes a client/server database—the MICROSOFT DATA ENGINE (MSDE)—on the installation CD-ROM. Although MSDE is a scaled down version of MICROSOFT's flagship database product, SQL SERVER, ACCESS 2000 continues to use its own JET database engine by default.



You are not expected to perform the exercises in this lesson. Instead, you should just sit on your hands and follow along.

9.1.2 The client-server environment

To set the stage for the demonstration, let us assume the following: Instead of storing your employee data in a payroll system built on top of dBASE IV files, you purchased the human resources module from an enterprise resources planning (ERP) vendor like ORACLE, PEOPLESOFT or SAP.



At their core, ERP systems are simply standardized applications that run on top of large, integrated relational database systems.

In this scenario, we are assuming that you want to by-pass the ERP application and read its underlying database directly. In this way, you can use up-to-date employee information in the order entry application that you are building.

9.1.2.1 Database server details

Assume that the relational DBMS being used by the ERP system is INTERBASE. Since INTERBASE is open-source software, it can be freely downloaded.



Typically, companies that spend tens of millions of dollars on ERP installations do not decide to run the whole thing on top of an open-source database they downloaded from the Internet (although this may be changing). I am using INTERBASE as an example because—who knows—you may want to download a copy and try the exercises in this lesson at some point in the future when you have a whole weekend to waste.

Before we connect to the database server, there are a handful of technical details we should know about the server itself:

- The database server is running INTERBASE version 6.0 on top of the LINUX operating system (downloaded from REDHAT).¹
- Assume that the IP address of the database server is `misux.bus.sfu.ca`.
- A **view** called `SALES_REPS` has been created to show only those employees classified as

¹ Rather than being an industrial-strength ERP platform, this machine is a crummy old Pentium 120 that I use for development purposes. Since it is an unwanted machine (without a monitor or a mouse) running an open-source DBMS on top of an open-source operating system, the total cost to build the database server was negligible.

sales people and to hide confidential information about pay levels.



Views and queries will be discussed in greater detail in [Lesson 10](#). Briefly, a view is like a “virtual table” that is used to filter and organize data from one or more database tables. For example, a view could provide the data in the **Employees** table sorted by last name. A second view could show the same data sorted by employee number. The important thing is that neither view changes the ordering or the structure of the data in the **Employees** table.

- An ODBC driver to communicate with the database server has been purchased and installed on the client machine. In this example, the INTERBASE ODBC driver has been provided by third-party company called EASYSOFT.



Although the high-level language used for creating tables and queries is standardized across all major DBMS vendors, the low-level languages and interfaces required for client/server communication are far from standardized. Consequently, a special ODBC driver is required to translate standard high-level commands into the



idiosyncratic low-level commands required by each DBMS.

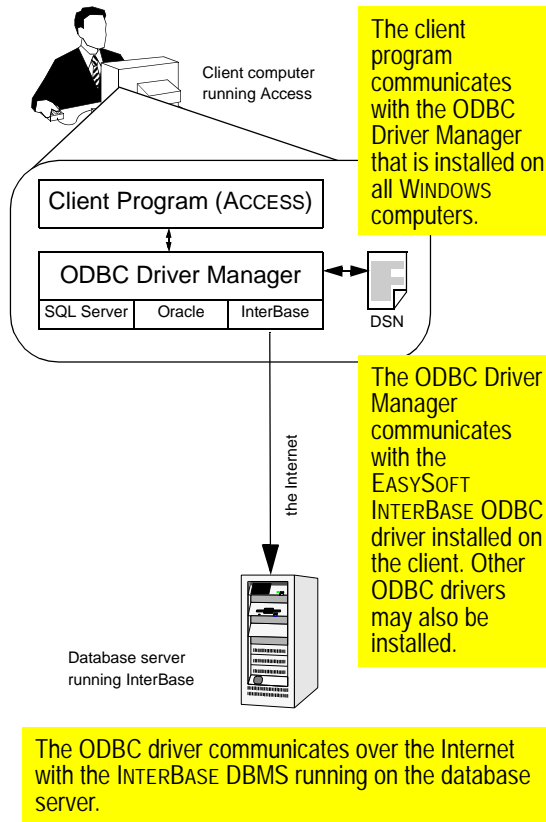
9.1.2.2 Setting up an ODBC connection

To get data from the database server to the client machine and into a program like MICROSOFT ACCESS, some infrastructure has to be in place:

1. The client computer has to know how to find the database server on the network.
2. The server needs some means of authenticating the client application as a user. Not just anyone should be able to log into the server and download payroll information.
3. Some form of network transport has to be provided so that my client computer can send the server instructions and the server can send back data.
4. The data received from the server must be reassembled into rows and columns.

Fortunately, all of this infrastructure is provided by the ODBC middleware. The ODBC standard is discussed in greater detail in [Section 9.4.1](#). At this point, you can think ODBC as an adapter that permits different systems from different vendors to be linked together, as shown in [Figure 9.1](#).

FIGURE 9.1: The role of ODBC software in a client/server connection.





9.2 Learning objectives

- see how an ODBC data source is configured in WINDOWS.
- understand how ACCESS can be used as a front-end to a client/server database
- see how easy it is to change the data source when using the ODBC infrastructure
- see how other WINDOWS applications can use an ODBC connection
- gain a basic understanding of what ODBC middleware is and how it is used

9.3 Exercises

9.3.1 Linking to a client/server database

To connect the database server, I must first define a "data source name" (DSN) on the client machine. The DSN simply permits me to specify and save important details concerning the connection, such as the server name, the user name that should be used to log into the server, and so on.

- ➔ First, I invoke the ODBC Data Source Administrator Tool from the WINDOWS Control Panel.

- ➔ I select the System DSN tab and press the **Add** button, as shown on the left-hand side of [Figure 9.2](#).



If you are wondering what all the tabs are for in "ODBC Data Source Administrator" window in [Figure 9.2](#), there are two different ways to save the DSN information for a particular database connection: in the WINDOWS registry ("Machine DSN") and to a small text file ("File DSN"). Machine DSNs can be further subdivided into those that can be seen and used by all users of a system ("System DSN") and those that are specific to a particular user ("User DSN"). The basic function of the DSN is the same in all cases, however.

9.3.1.1 Specifying the data source details

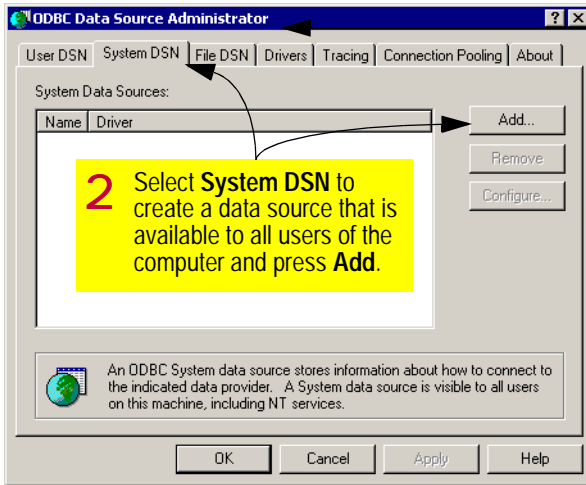
- ➔ I select a driver that is designed to communicate with the DBMS running on the server. In this case, I select the EASYSOFT INTERBASE version 6.0 driver, as shown on the right-hand side of [Figure 9.2](#).



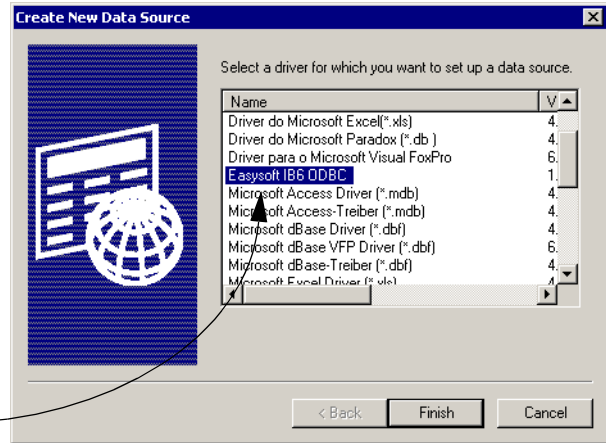
From a pricing point of view, there are three types of ODBC drivers: free, expensive, and insanely expensive. ACCESS automatically installs free ODBC drivers for a number of database systems,



FIGURE 9.2: Create a new data source name (DSN) for the database server.



1 Open the ODBC Administrator from the WINDOWS control panel.



3 Scroll through the list of ODBC drivers installed on the client machine and find the ODBC driver for INTERBASE.

including MICROSOFT SQL SERVER and ORACLE (although you need a client license from ORACLE to use the ODBC driver). Prices for third-party drivers can run into the hundreds of thousands of dollars (I am not kidding). The EASYSOFT driver retails for about US\$100 and several open-source

drivers for INTERBASE are under development.

➔ I click the **Finish** button to exit the ODBC Data Source Administrator Tool.



Next, the configuration tool for the EASYSOFT driver takes over from the ODBC Data Source Administrator. The EASYSOFT dialog collects additional information that is specific to INTERBASE database servers.

➔ First, I enter a name for the DSN and a description. Then I type in the Internet address of the database server (**misux.bus.sfu.ca**) and the location of the target database on the server (**/home/brydon/IBData/Payroll.gdb**), as shown in [Figure 9.3](#).

➔ Then, I enter a valid user name and password.



Like all client/server databases, INTERBASE has a robust security infrastructure. The "2NP" account used in this example has been granted read-only access to the **SALES_REPS** view, but no access to other, more sensitive information such as that contained in the **PAY_EMPS** table.

➔ Finally, I press the **Test** button to make sure I have entered the DSN information correctly. The good news is shown in [Figure 9.4](#).

Since the test was successful, I have a working ODBC connection to a remote database.

FIGURE 9.3: Enter the InterBase-specific information required to make a connection to the server.

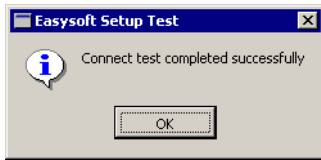
1 Enter the location of the database (including the IP address of the server and the location of the database file on the server).



The nice thing about using the Internet protocol (TCP/IP) to identify the location of the server is that the server can be



FIGURE 9.4: Test the ODBC connection to ensure it works.



anywhere in the world. As long as I have the server's IP address (in this case, **misux.bus.sfu.ca** or **142.58.223.133**), I can access the data.

9.3.1.2 Using the ODBC connection to create a linked table

Now that an ODBC link is in place, it can be used by any program that supports the ODBC standard, such all MICROSOFT OFFICE applications, high-end statistics packages, even the shipping software provided by many courier companies. In my case, I want to use the ODBC connection to create a linked table in ACCESS.

➔ I return to ACCESS and initiate the linked table dialog as you did in [Section 8.3.3](#).

➔ Instead of selecting a specific file type in the "Link" dialog, I select "ODBC Database".

When I select an OBDC data source, a list of both "machine" and "file" data sources pops up.

➔ I select the **Machine Data Source** tab and find the DSN of the payroll system connection I set up earlier, as shown in [Figure 9.5](#).

If everything goes well, ACCESS will use the ODBC link to request a list of tables and views in the payroll database. As [Figure 9.6](#) shows, some of the tables are system tables (prefixed by **RDB\$**). System tables contain data used by the DBMS itself and are of little interest to us.

➔ I select the **SALES_REP** view, as shown in [Figure 9.6](#).

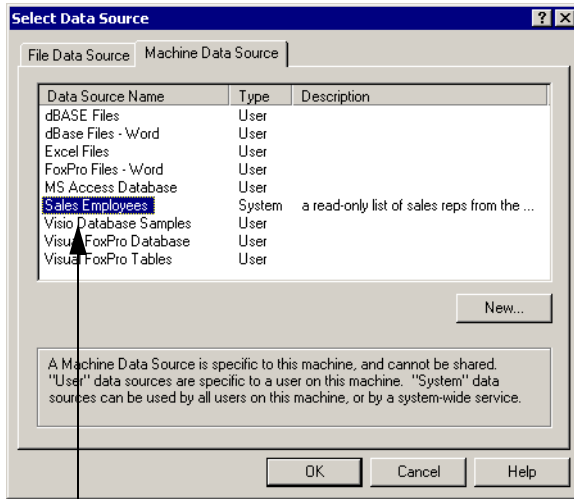
Unlike tables, views do not have a primary key defined. Thus, I am asked to select one field (e.g., **EMP_ID**) as the "unique record identifier" for the purpose of the linked table, as shown in [Figure 9.7](#).



Note that the icon for the ODBC linked table is different than that of the other tables in the ACCESS database.



FIGURE 9.5: Use the ODBC connection as the data source for the linked table.

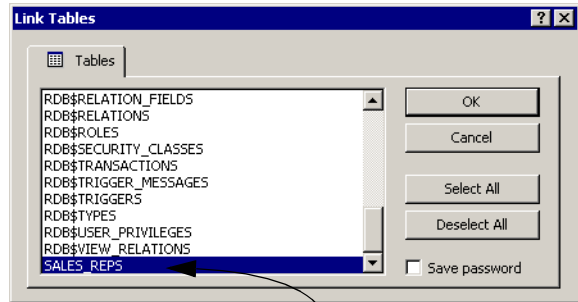


1 Select **Machine Data Source** and find the payroll system I set up in the previous section.

➔ To demonstrate the security features of the linked table, I open it in data sheet mode and make a change to one of the records. As Figure 9.8 shows, the "2NP" account is permitted to view, but not change the data on the server.

FIGURE 9.6: Create links to one or more tables in the remote database.

! The ODBC driver returns all the tables in the database, including the system tables.



1 Select the **SALES_REP** table (which is actually a view) and press OK.

! If the database administrator were to grant modify privileges to the "2NP" account, the ACCESS application could be used to add, modify, and delete data from the remote database. Although such functionality does not make sense in the context of payroll data, there are other situations in which an ACCESS front-end to



FIGURE 9.7: Select a field to be the unique record identifier for the *SALES_REP* linked table.

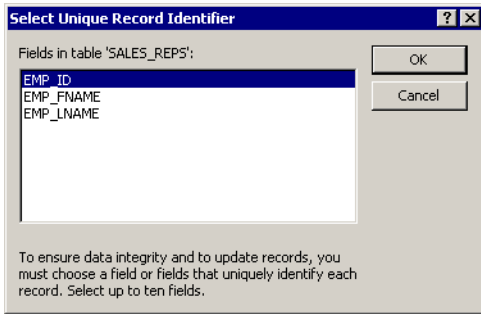
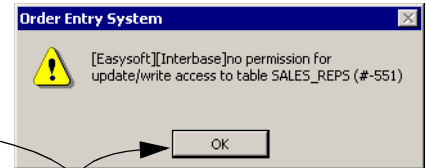
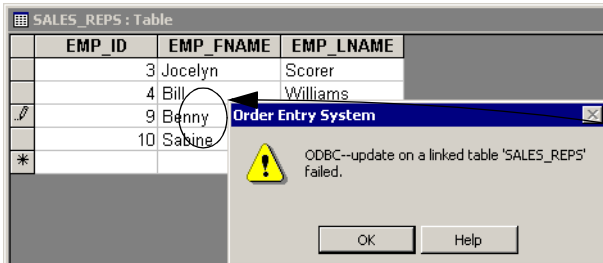


FIGURE 9.8: User "2NP" does not have update permission on the *Employees* table.



Multiple error messages are received when the "2NP" user account attempts to make a change to the data.



a client/server database is extremely convenient.



Accessing data over a network is bound to be slower than accessing a local copy. In some cases, network delays or unreliability make client/server access infeasible. In such situations, data **replication** and **synchronization** become very attractive technologies.

9.3.2 Changing the target database

To illustrate the power of middleware, assume that you have built additional ACCESS objects (such as queries, reports, and forms) on top of the **SALES_REPS** linked table. Assume as well that the people in charge of the ERP system decide to switch from an INTERBASE DBMS to MICROSOFT SQL SERVER on a different machine. In this section, I will demonstrate how easy it is to incorporate this change into an ACCESS application.



[Show me](#) (lesson9-1.avi)

- ➔ First, I create a new DSN. However, instead of using the INTERBASE driver, I use the SQL SERVER driver that ships with ACCESS.
- ➔ I fill in the SQL SERVER-specific dialog boxes provided by the ODBC driver required to

fully specify the DSN, as shown in [Figure 9.9](#) and [Figure 9.10](#).



As [Figure 9.9](#) illustrates, each ODBC driver requires slightly different DSN information. For example, SQL SERVER supports multiple network protocols and therefore requires that the desired network protocol be specified in the DSN.

- ➔ I delete the existing linked table (which, of course, has no effect whatsoever on the table stored on the database server) and create a new link using the SQL SERVER DSN, just as in [Section 9.3.1.2](#).

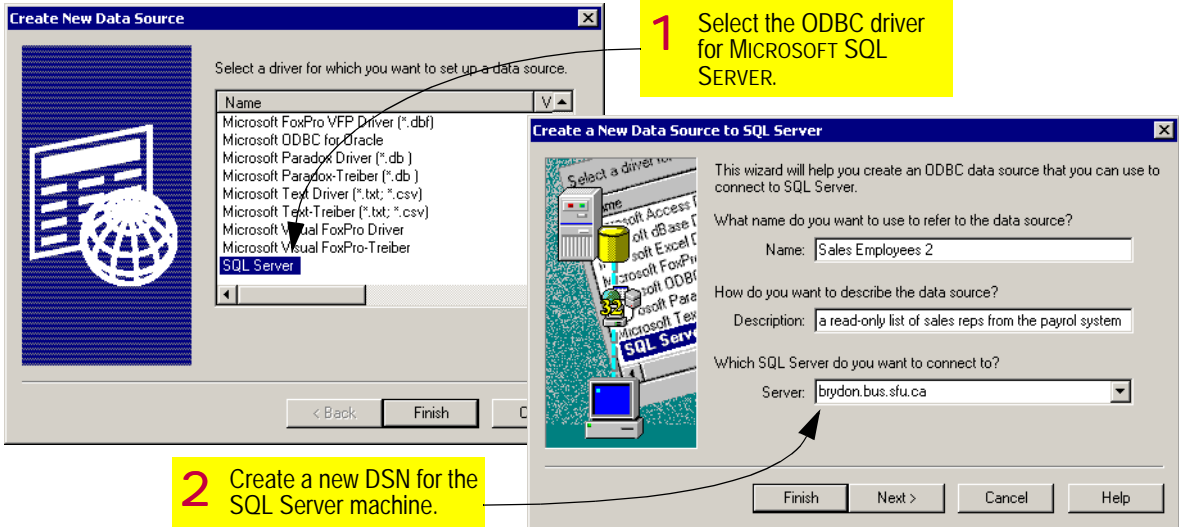
At the end of the process—which is only superficially different from the INTERBASE example—I have a linked table with the same data as before. In most cases, no additional changes are required to my ACCESS application—that is, all my queries, forms, reports, and VISUAL BASIC programs work just as they did when I was running INTERBASE off a LINUX machine.

9.3.3 Changing the client application

Once an ODBC data source is set up on the client computer, it is possible for other applications to use the same connection. In this section, I will use an ODBC connection to bring employee information into MICROSOFT EXCEL. It is important to note that the techniques shown



FIGURE 9.9: Create a new DSN using the SQL SERVER ODBC driver (part 1)



below can be used to bring data into WORD (e.g., for a mail merge) or virtually any other WINDOWS program.



[Show me](#) (lesson9-2.avi)

➔ I start by opening a new worksheet in EXCEL and selecting **Data** → **Get External Data** → **Create New Query** from EXCEL'S main menu.

Rather than having their own functionality for handling ODBC connections, all OFFICE applications except ACCESS rely on a separate program called MICROSOFT QUERY to do the dirty work.



Like many of the helper and add-in programs in OFFICE, MICROSOFT QUERY may or may not have been installed when you installed OFFICE. If you get an error message similar to the one shown in



FIGURE 9.10: Create a new DSN using the SQL SERVER ODBC driver (part 2)

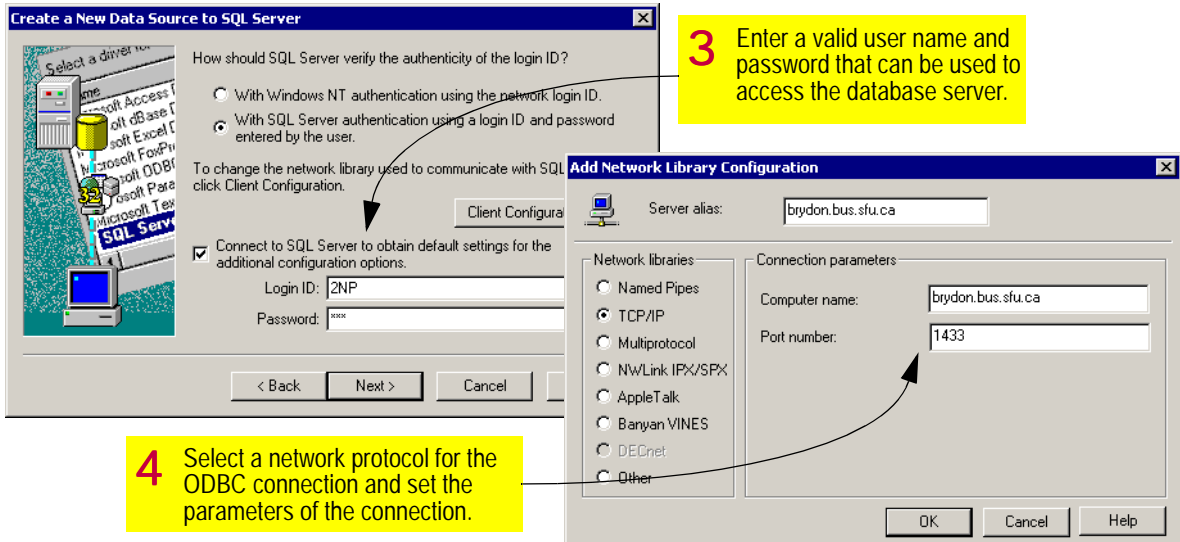


Figure 9.11, you must re-run the setup program from the OFFICE CD-ROM, as shown in Figure 9.12.

Assume that I am using EXCEL 97 and that I have created a "file DSN" that is identical in every way to the "system DSN" created in Section 9.3.1.1 (except that file DSNs are saved to a text file rather than to the WINDOWS registry).

- ➔ From within MICROSOFT QUERY, I select the file DSN called **Sales Employees 3** from the list, as shown on the left-hand side of Figure 9.13.
- ➔ I then expand a table and select the columns I want to include in my spreadsheet, as shown on the right-hand side of Figure 9.13.



FIGURE 9.11: MICROSOFT QUERY is not installed on the computer.

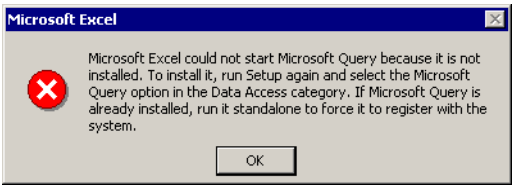
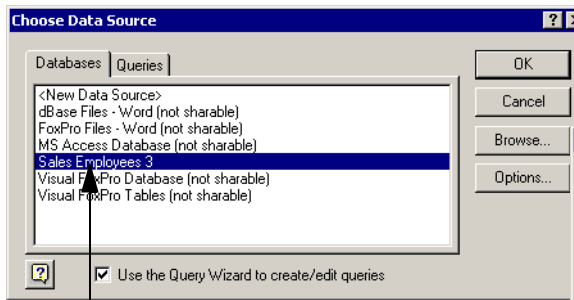


FIGURE 9.13: Use the INTERBASE ODBC connection to bring data into MICROSOFT QUERY.



1 Create a file DSN that links to the INTERBASE server. Save it as **sales Employees 3**.

2 When prompted by MICROSOFT QUERY for a data source, select the **sales Employees 3** DSN.

3 Select the table (or view) and the fields that you want to be shown in the spreadsheet.

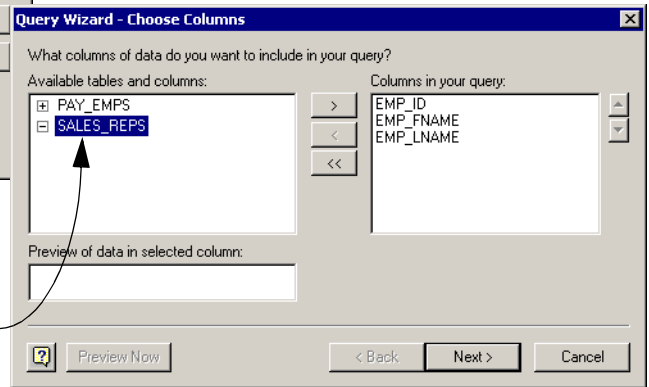




FIGURE 9.12: Re-run the OFFICE setup program to install MICROSOFT QUERY.

1 Re-run the OFFICE setup program and select **Add/Remove**.

2 Select the "Data Access" entry and press **Change Option**.

3 Ensure MICROSOFT QUERY is checked and press **OK** to install the component

Microsoft Office 97 Setup

Welcome to the Microsoft Office 97 installation maintenance program.

This program lets you make changes to the current installation. Click one of the following options:

- Add/Remove...** Add new components or remove installed components from the current installation.
- Repeat the last installation to restore missing files

Microsoft Office 97 - Maintenance

In the Options list, select the items you want installed; clear the items you want to be removed. A grayed box with a check indicates that only part of the component will be installed. To select all components in the Option list, click Select All.

Options:

Options:	Description:
<input type="checkbox"/> Microsoft Binder 1605 K	Database Drivers and utilities that let you connect to data in a variety of different formats.
<input checked="" type="checkbox"/> Microsoft Excel 21875 K	
<input type="checkbox"/> Microsoft Word 31074 K	
<input type="checkbox"/> Microsoft PowerPoint 30781 K	
<input checked="" type="checkbox"/> Microsoft Access 44180 K	
<input type="checkbox"/> Microsoft Outlook 26417 K	
<input type="checkbox"/> Web Page Authoring (HTML) 7194 K	
<input type="checkbox"/> Microsoft Bookshelf Basics 125 K	
<input checked="" type="checkbox"/> Data Access 7787 K	
<input checked="" type="checkbox"/> Office Tools 2904 K	

Change Option... Select All

Folder for Currently Selected Option: C:\Program Files\Microsoft Office 97\Office\Library\MSQuery Change Folder...

Space required on C: 77319 K Components to Add:
Space available on C: 999999 K Components to Remove:

Continue Cancel

Microsoft Office 97 - Data Access

In the Options list, select the items you want installed; clear the items you want to be removed. A grayed box with a check indicates that only part of the component will be installed. To select all components in the Option list, click Select All.

Options:

Options:	Description:
<input type="checkbox"/> Database Drivers 1285 K	An application that helps you retrieve data from external data sources for use in Microsoft Excel or Word.
<input checked="" type="checkbox"/> Microsoft Query 1418 K	
<input checked="" type="checkbox"/> Data Access Objects for Visual Basic 3298 K	

Change Option... Select All

Folder for Currently Selected Option: C:\Program Files\Microsoft Office 97\Office\Library\MSQuery Change Folder...

Space required on C: 4716 K Components to Add: 1
Space available on C: 999999 K Components to Remove: 0

OK Cancel



Unlike ACCESS, MICROSOFT QUERY hides the system tables (e.g., **RDB\$CHECK_CONSTRAINTS**) by default.

MICROSOFT QUERY asks me if I want to filter and sort the data and where I want the data to be placed in the spreadsheet.

- ➔ I select a location in the spreadsheet and the data magically appears.
- ➔ To make sure I have the most up-to-date data from the database, I press the **Refresh Data** button on the “External Data” toolbar, as shown in [Figure 9.14](#).

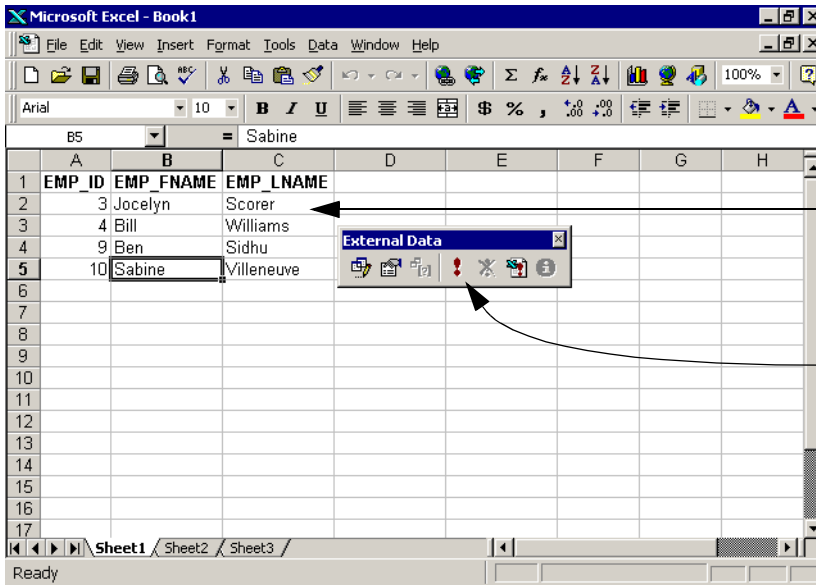


FIGURE 9.14: Data from the database server is copied into the spreadsheet.

1 Specify the location within the EXCEL worksheet for the data returned by MICROSOFT QUERY.

2 Press the **Refresh Data** button to re-query the source database.

? Changes to the database are not automatically noticed by the ODBC connection. Hence the need to refresh.

The flow of information from the database to the spreadsheet is one-way. That is, if a change is made to the payroll system, the changes to

the data to appear in the spreadsheet (once the **Refresh Data** button is pressed). However, if the data in the spreadsheet is changed, none of



the changes are saved back to the database. Moreover, any changes to the data in the spreadsheet are overwritten the next time the data is refreshed.



With a surprisingly small amount of programming within EXCEL, it is possible to use the ODBC connection to update the client/server database from the spreadsheet. However, given the lack of structure and control in spreadsheets, it is not clear that using EXCEL as a two-way front-end to your data is a good idea. In the case of payroll data that is feeding an ERP system, anything other than read-only access is a very bad idea.

9.4 Discussion

9.4.1 ODBC and client/server databases

If you accept the premise that data is an organizational resource, then the notion of many users working with (and hoarding) their own personal copies of data on their desktops is bound to be troubling. And so it should be.

An elegant way around this centralization/decentralization dilemma is to use ACCESS as the front-end to tables stored on a centrally maintained, industrial strength, multi-user DBMS. In this way, users can continue to work

with ACCESS as if the tables were stored locally. However, in reality, they are looking at and interacting with the same data as everyone else in the organization.

ODBC (open database connectivity) is a MICROSOFT-developed standard that provides a common interface to virtually all databases. The key to ODBC is drivers that translate basic database query commands into commands understood by the particular data source at the other end of the wire.

As long as you have the correct ODBC driver and can generate ODBC-compliant commands, you really do not have to know anything about the source of the data—it could be ORACLE, it could be MICROSOFT SQL SERVER, it could be one of hundreds of other types of data storages systems.

There is plenty of information on ODBC on the MICROSOFT web site. ORACLE, on the other hand, prefers to more or less deny the existence of ODBC. Not surprisingly, there are a number of third-party vendors of ODBC software that neatly bridge the ideological chasm between MICROSOFT and other database vendors such as ORACLE. For example, both MERANT (formerly INTERSOLV) and Vancouver's own SIMBA provide middleware products that take care of many of the frustrating client/server networking issues



as well as provide ODBC → native database translation.



MICROSOFT is in the process of superseding ODBC with new standards (e.g., OLE DB and ADO). Check the MICROSOFT web site for the latest information on this ever-changing alphabet soup of data-access standards and middleware. You will gain some experience with ADO in [Lesson 28](#).

including software-on-demand, distributed databases, distributed computation (e.g., SETI@Home), interactive gaming, and the list goes on and on.

9.4.2 The M2M Internet

Much of the discussion about the Internet to date has been around the use of browsers to surf the world wide web (WWW).¹ Business models based on the web—such as “business to consumer” (B2C)—rely on people communicating with machines over a network that is essentially free and ubiquitous (at least in some parts of the world).

In this lesson, you have seen a different use of the Internet: machine-to-machine (M2M) communication over the same TCP/IP networks that carry the traffic from million of web surfers. The potential of machine-to-machine communication over a wide-area network such as the Internet is huge: many things are possible

¹ Indeed, “surfing the web” is so common that the term has ceased to be regarded as a mixed metaphor.