PART I
Introduction

- **CHAPTER 1:** Self-Service Business Intelligence and Microsoft PowerPivot
- **CHAPTER 2:** A First Look at PowerPivot

1

Self-Service Business Intelligence and Microsoft PowerPivot

WHAT'S IN THIS CHAPTER?

- Reviewing SQL Server 2008 R2
- Understanding Self-Service Business Intelligence
- Getting to know PowerPivot
- Taking a look at PowerPivot applications
- Taking a look at PowerPivot for Excel
- Taking a look at PowerPivot for SharePoint
- Taking a look at the VertiPaq engine

PowerPivot is Microsoft's entry into the self-service business intelligence (BI) arena. PowerPivot was built with specific goals in mind, and this chapter will explain some of those goals. PowerPivot was also specifically designed not to address certain goals, and this chapter will also discuss those decisions as well. Some dependencies PowerPivot had on other groups and technologies (specifically, Microsoft Office, especially Excel and SharePoint) led to how it was designed and built. This chapter will explore those goals, dependencies, and decisions.

By the end of this chapter you will have a clear idea of the "what and why" of PowerPivot. Subsequent chapters will go into much greater detail on how to work with PowerPivot, and describe its features with the goal of helping you become a professional PowerPivot user who can get the most out of this innovative product.

SQL SERVER 2008 R2

PowerPivot is included in the R2 release of SQL Server 2008. The "R2" in the name might give you the impression that this release of SQL Server is a minor update to SQL Server 2008. If you thought that, you would be wrong. The 2008 R2 release includes major new functionality, including the following:

- Application and Multi-server Management capabilities, which provide the ability to manage a data environment that includes many servers
- Stream Insight, which supports building applications that do high-volume, complex event processing
- Master Data Services, which helps organizations manage and standardize their enterprise data across applications and systems.
- PowerPivot, Microsoft's self-service BI offering, which is the subject of this book.

SQL Server 2008 R2 was designed to be a BI-centric release of SQL Server, with a particular focus on self-service BI.

SELF-SERVICE BUSINESS INTELLIGENCE

If you ask different people to define *self-service business intelligence* (or *self-service BI*), chances are you will get different answers, depending on who you ask. That's because self-service BI is a new BI paradigm that is still being defined and created. It has not been around long enough to be standardized in the way that other paradigms like relational databases or even traditional BI has. And yet, it is also not an approach that is starting completely from scratch.

Many of the concepts in self-service BI sprang from earlier BI principles and practices. This book refers to those earlier BI paradigms as "corporate BI." Self-service BI, then, is something new, but it's also based on some things that came before. To help explain the relationship between corporate BI and self-service BI, consider another technological advancement that is familiar to many people — the move from command-line interfaces (CLIs) to graphical user interfaces (GUIs) in computer operating systems.

Earlier personal computer operating systems (such as MS-DOS) provided a very simple interface to users: the command line. This interface was text-based, as opposed to graphics-based, and allowed you to type in one line at a time. The response you got back was a single line of text, and perhaps a change in the state of the system that wasn't visible to you (the command-line user).

There were, of course, ways to get more friendly and capable applications on those operating systems, but in order to do it, you had to build everything yourself. The operating system didn't provide standardized components like graphical controls, easy-to-use interfaces to the file system, or a common way to talk to devices like printers. In that world, mere mortals (those without detailed computer knowledge and low-level programming skills) would need custom applications built for them in order to work with computers.

Back then, as you can imagine, the majority of people did not see the computer as an integral part of the way they did their jobs. People who could build the custom applications needed to make

computers a part of the way people did their work were few and far between. Since you had to build all the functionality your application needed, applications took a long time to build. Even if you had an idea of how a computer application could help you do your job, unless you had the money to hire someone to build it, or had that highly specialized knowledge of how to implement it yourself, you wouldn't be able to realize your idea of a computer application that could help you do your job.

With the emergence of GUIs, operating systems provided a much richer set of common functionality that applications could make use of without having to implement all the low-level details themselves. For example, instead of having to write your own printer drivers for every printer you wanted to support in your application, you could simply rely on the printer drivers that were provided by the operating system. When coupled with new application-creation tools (such as Visual Basic), that allowed more people with less detailed knowledge and skill to build the applications that people wanted and needed in order to get their jobs done. Then you had the ingredients necessary to make computers an integral part of more and more people's daily lives. Many more people than before could realize the ideas they had about how computers could help them do their work.

GUIs, and the operating systems that supported them, were a completely new paradigm of how people interacted with computers. And yet, underneath were many of the same components that were there before GUIs came on the scene. They extended, augmented, and standardized what came before and allowed much greater capability for a larger variety of people than their predecessor, the command-line-based operating system.

Self-service BI aims to effect the same sort of paradigm shift in the BI world that modern operating systems did for general computer users. Here is what the state of BI looked like before the self-service concept:

- In order to build BI applications, you had to be a BI developer with highly specialized skills, or have enough money (or clout) to hire one. BI applications were generally custom-built.
- Once your BI application was built and deployed, it could be difficult to change in response to a change in the business situation or customer requirements.
- If you worked in the data center and were responsible for BI applications, chances were that you had to maintain every application in a separate way. Each one may have had its own special requirements for deployment, maintenance, backup, and so on.
- If you were an analyst needing to use data to make your business decisions but were not able to build the BI application you needed, you might have cobbled together data from various sources in an ad-hoc way in order to do your analysis. You probably used spreadsheets to do this. Once you did, your application and its data moved out of the realm of the corporate BI systems and into the wild and wooly world of desktop and laptop systems.
- As a result, your analytical data might have become disconnected from its source and, as a result, outdated as the source data changed. Refreshing your data, when it was possible, could be difficult and time-consuming.
- Since your data now lived in a spreadsheet file, when you shared it, you lost explicit control over it. If all or part of the data was confidential, you couldn't prevent those you shared it with from sharing it with unauthorized people.

Contrast that situation with Microsoft's vision of self-service BI, which it calls "managed self-service business intelligence," as implemented in PowerPivot:

- Anyone can easily build his or her own self-service BI applications using tools that they already know, starting with Microsoft Office Excel.
- Self-service BI applications are easy to update and modify. This can be done by the person who built them, even if that person isn't a BI application developer.
- If you work in the data center and are responsible for deploying and managing self-service BI applications, you can manage all your published applications in a common way. You have the tools you need to track usage, administer security, and deploy new hardware in response to the needs of the system.
- Your analytical data remains connected to its source. Refreshing your application from source data is easy, and can even be done automatically by the system.
- You can easily share data, but in a controlled way. Customers of your application can access it over the Web (internal or external) without needing anything other than a Web browser.

Microsoft's previous tag line for its BI products was "business intelligence for the masses." With self-service BI, that tag line can now be expanded to "business intelligence for the masses, by the masses." The Microsoft product that aims to make this possible is PowerPivot.

POWER PIVOT: MICROSOFT'S IMPLEMENTATION OF SELF-SERVICE BI

PowerPivot is made up of two separate components that work together:

- *PowerPivot for Excel* — PowerPivot for Excel is an Excel add-in that enhances the capabilities of Excel, enabling business analysts and Excel power users to create and edit PowerPivot applications.
- *PowerPivot for SharePoint* — PowerPivot for SharePoint extends Microsoft Office SharePoint Server to include the capabilities to share and manage the PowerPivot applications that are created with PowerPivot for Excel.

The next section describes what a PowerPivot application looks like.

PowerPivot Applications

At first glance, PowerPivot applications look just like Excel workbooks. And that they are, but they also include something more — PowerPivot data and metadata embedded in the workbook itself. This allows a PowerPivot-enhanced Excel workbook to contain much more functionality than can be contained in a regular Excel workbook that doesn't connect to external data sources. For example, PowerPivot workbooks can contain tables that are much bigger than Excel tables. Excel tables (in Office 2007 and beyond) can contain 1 million rows of data. PowerPivot tables inside an Excel workbook can contain tens or even hundreds of millions of rows of data, as shown at the bottom of Figure 1-1.

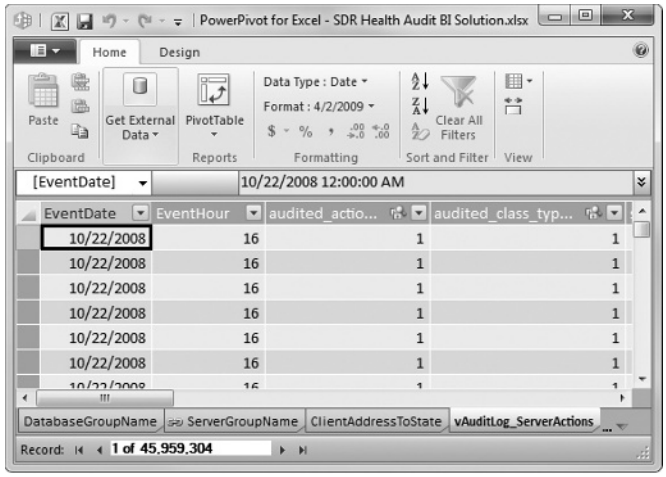

FIGURE 1-1: A PowerPivot table with more than 1,000,000 rows of data

The PowerPivot tables in the workbook make up the PowerPivot data mentioned previously. These tables can be joined together and then used as the source data for Excel PivotTables and PivotCharts, which can then be used for analysis and reporting.

PowerPivot applications can be shared among stakeholders, co-workers, management — anyone who wants to view or interact with them. To do this, you publish your PowerPivot workbook to Microsoft Office SharePoint Server. People can then browse and/or interact with your application using either the Excel client or a Web browser. You can also set up PowerPivot to automatically refresh your application from the source data either once or on a regular schedule.

To summarize, you can think of a PowerPivot application as an Excel workbook "on steroids." It gives you all the power of Excel, plus the greater analytical capability necessary to deliver true self-service BI.

Now, let's take a look at the two components that make PowerPivot applications possible.

PowerPivot for Excel

PowerPivot for Excel is the tool you use to create and edit PowerPivot applications. It supports integrating data from various external data sources, enriching that data with custom calculations and adding relationships between tables, as well as using that data to do analysis in Excel using features such as PivotTables and PivotCharts. PowerPivot for Excel is implemented as a managed Excel add-in that provides the user interface for working with PowerPivot data. Figure 1-2 shows the architecture of PowerPivot for Excel.

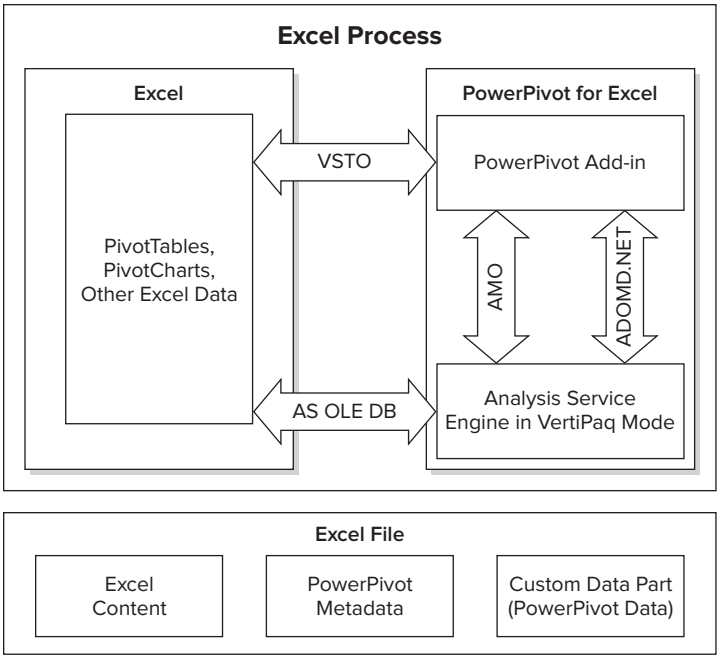

FIGURE 1-2: The architecture of PowerPivot for Excel

PowerPivot for Excel also includes the VertiPaq engine, a local, in-process version of the Analysis Services engine in VertiPaq mode (which is discussed later in this chapter). The PowerPivot for Excel add-in communicates with the VertiPaq engine via the traditional Analysis Services interfaces Analysis Management Objects (AMO) and ActiveX Data Objects Multi-Dimensional (ADOMD.NET). The add-in communicates with Excel via its object model using the Visual Studio Tools for Office (VSTO) managed interface. Excel communicates with the in-process VertiPaq engine via the Analysis Services OLEDB provider.

When you are working with PowerPivot for Excel, the PowerPivot data will reside in memory. But when you save your workbook, PowerPivot will store its data and metadata inside the Excel file, as shown in Figure 1-2. The in-memory database will be stored in a section of the file called the Custom Data Part (CDP). The writing of the CDP is done through a public interface that first appeared in Excel 2010. It allows applications to write and retrieve their own data inside an Excel file.

PowerPivot for Excel will also store metadata and workbook settings in XML streams inside the Excel file. This saved metadata allows PowerPivot to attempt to reconstruct a workbook's data model if the CDP data becomes corrupted. If the structure is successfully recovered, you may be able to refresh the workbook's external data, and recover the contents of the workbook.

PivotTables and PivotCharts are the main analytical tools in Excel. The PowerPivot add-in and VertiPaq engine work with Excel to provide you with the capability to use these tools to do your self-service BI analysis.

Microsoft wants to make it as easy as possible to get started with PowerPivot, so it is making this part of PowerPivot available as a free download on the Web. PowerPivot for Excel has the following prerequisites:

- *.NET 3.5 SP1* — Installation of this component is not needed on later operating systems like Windows 7, which includes it as part of the operating system itself. If you are installing on an older operating system such as Windows XP or Windows Vista, you will need to install .NET 3.5 SP1. Install this before installing Office 2010.
- *Excel 2010 + Office Shared Features* — PowerPivot for Excel requires Excel 2010. It will not install on earlier versions of Excel. Also, the architecture of PowerPivot for Excel must match the architecture of Excel itself. If you have 32-bit Excel installed, you must install the 32-bit version of PowerPivot for Excel. If you have 64-bit Excel installed, you must install the 64-bit version of PowerPivot for Excel.

When installing Office 2010, you must also install the Office Shared Features item along with Excel. This is because PowerPivot for Excel is a Visual Studio Tools for Office (VSTO) add-in and requires the VSTO run-time in order to work. Office Shared Features will install the VSTO run-time. If you install Excel without Office Shared Features, you will have to uninstall Excel and then re-install including Office Shared Features.

- *Platform Update for Windows Vista/Windows Server 2008* — PowerPivot for Excel requires this component if you are running on the Windows Vista or Windows Server 2008 operating systems. You can find more information about this prerequisite at `http://support.microsoft.com/kb/971644`.

Note that this component is installed via Windows Update. Also note that the Platform Update for Windows Vista/Windows Server 2008 is an important, rather than a critical, update. If you have set up Windows Update to only install critical updates, you may miss this.

- *Drivers for connecting to non-Microsoft data sources* — If you want to import data from data sources other than Microsoft data sources that are included with PowerPivot (such as Oracle, Teradata, or DB2), you must acquire and install those drivers and any related

client components yourself. They are not included with PowerPivot for Excel. Importing data and more details on the data sources that are supported by PowerPivot will be covered in Chapter 3.

Note that there are also operating system requirements, as shown in Table 1-1.

TABLE 1-1: PowerPivot Operating System Requirements

OPERATING SYSTEM	REQUIREMENT
Windows XP	SP3 or greater, 32-bit only
Windows Server 2003 R2	32-bit or on WOW64 mode on 64-bit only
Windows Vista	SP2 or greater
Windows Server 2008	SP2 or greater
Windows 7	No special requirements
Windows Server 2008 R2	No special requirements

After you install the necessary prerequisites and PowerPivot for Excel, launch Excel. The first time you launch Excel after installing PowerPivot for Excel, you will see a dialog asking for confirmation that you want to install the add-in. Accept this dialog, and PowerPivot for Excel will load. You will notice that the Excel toolbar now has one new tab called PowerPivot, as shown in Figure 1-3. This tab is your entry point into PowerPivot for Excel.

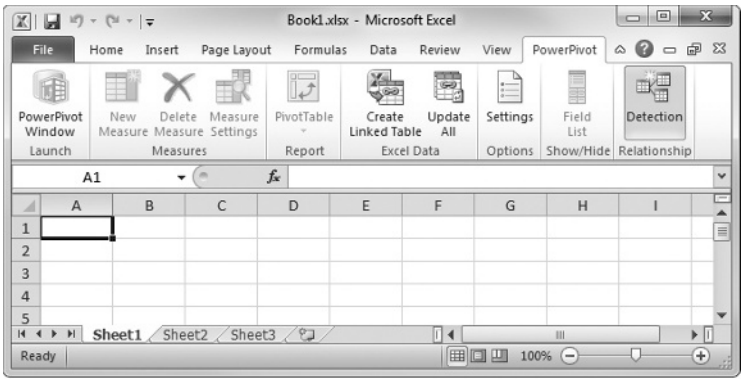

FIGURE 1-3: The PowerPivot tab in the Excel ribbon

If you click the PowerPivot Window button on the left side of the ribbon, the PowerPivot Window appears, as shown in Figure 1-4.

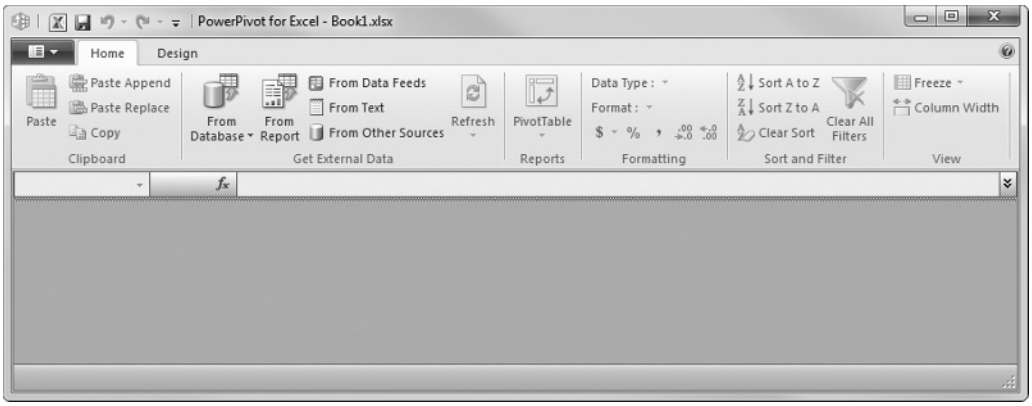

FIGURE 1-4: The PowerPivot Window

This is where you will import and work with analytical data in your PowerPivot application. You can think of the PowerPivot Window as the window into the PowerPivot data that is stored inside the workbook. Here is where you launch the import wizard, the tool that lets you import data from various data sources. These data sources include the following:

- Various relational data sources
- Data from Analysis Services and other PowerPivot workbooks
- File-based data from Microsoft Access, Microsoft Excel, and delimited text files
- Data feeds, which are syndicated sources of data that can be updated in a manner similar to RSS or Atom, including SharePoint 2010 lists and SQL Server Reporting Services reports (from SQL Server 2008 R2 and beyond)

You can also paste data from the Windows clipboard into a PowerPivot table if the contents of the clipboard are recognized by PowerPivot as tabular data. In addition, PowerPivot has the capability to create a PowerPivot table whose source data is a table in the Excel workbook. These two tables are linked such that if the Excel table is updated, the PowerPivot table will be updated as well, and the data will match. Chapter 3 goes into greater detail on all these methods of bringing data into PowerPivot.

As shown in Figure 1-2 earlier in this chapter, data imported into a PowerPivot workbook is stored inside the workbook itself. PowerPivot is capable of doing this because of a new feature in Excel 2010 that allows external applications (like Excel add-ins) to store custom data inside the Excel workbook. This is one of several reasons that PowerPivot requires Office 2010. (See the sidebar "Why PowerPivot Requires Office 2010.")

WHY POWERPIVOT REQUIRES OFFICE 2010

When the PowerPivot team made the decision to require Office 2010, influencing that decision were the features that were new to the 2010 release. These include the following:

- *The capability to embed custom data inside an Excel 2010 workbook file* — This allows PowerPivot data to be included in the PowerPivot workbook.
- *Slicers* — Slicers are a new type of control in Excel 2010. These controls make it much easier to filter analytical data in PivotTables and PivotCharts than in previous versions of Excel. PowerPivot for Excel includes enhancements to slicers over and above what is available in Excel without PowerPivot. Later chapters will examine this PowerPivot feature in more detail.
- *The Shared Service support in SharePoint 2010* — SharePoint 2010 has defined extensibility points that allow third-party applications to plug into the SharePoint infrastructure in a well-defined way that wasn't available in previous versions of SharePoint. In essence, they allow third-party applications to integrate with SharePoint in a way that is similar to the way Excel Services or SharePoint Search are integrated into SharePoint.

Another aspect of Office 2010 that PowerPivot benefits from is the availability of a 64-bit version. Since PowerPivot works with its data in-memory rather than from disk as in previous versions of Analysis Services (this aspect of PowerPivot is also examined in this chapter), the increased memory address space available to a 64-bit application is a real benefit when your analysis data gets large.

The PowerPivot team understood that requiring Office 2010 was a trade-off, and could limit (in some cases) the adoption of PowerPivot, but the team's eyes were on the future and on the increased capabilities that the latest version of Office enabled. The vision they had for PowerPivot required this.

After data is imported into PowerPivot, it appears in the PowerPivot Window as tables. Figure 1-5 shows the PowerPivot Window after a number of tables have been imported.

Tables in the PowerPivot Window appear similar to worksheets in an Excel workbook, but there is a difference. The table grid in the PowerPivot Window will only show the cells that include data. Also, every table gets its own sheet. In PowerPivot, the basic unit you work with is the table.

Another difference between PowerPivot data and data in Excel spreadsheets is that you can define relationships between tables. This allows much more powerful analysis than is possible in Excel worksheets using `VLOOKUP`. Relationships in PowerPivot and how to work with them will be discussed in Chapter 4, which will show how to work with PowerPivot data to build your self-service BI applications.

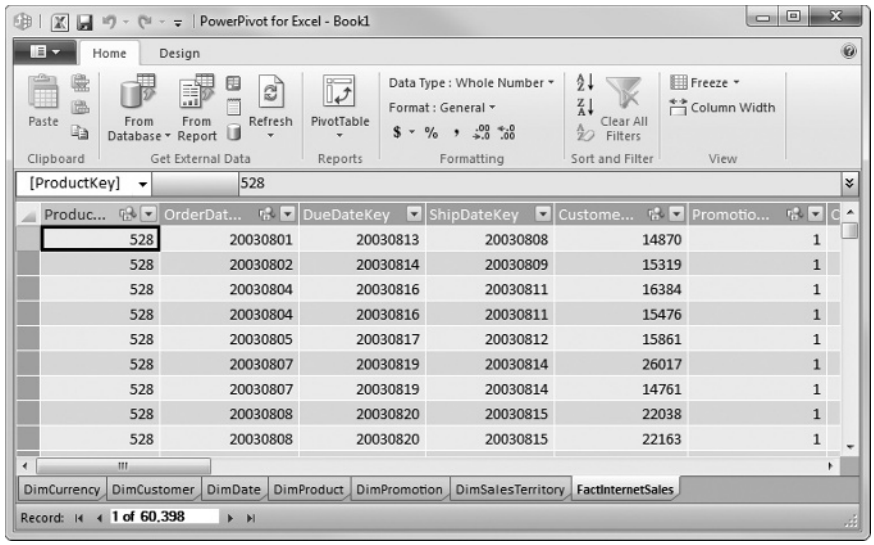

FIGURE 1-5: The PowerPivot Window with multiple tables imported

Yet another difference between PowerPivot and Excel is the sheer amount of data that can be included in a single table. In later versions of Excel, the number of columns and rows that can be included in a table was increased greatly to be able to have worksheets with up to 1 million rows of data. While this may seem like a lot, in the BI world, a table with 1 million rows of data is considered small or medium-sized. PowerPivot allows you to have tables that include tens or even hundreds of millions of rows of data.

Because PowerPivot is a part of Excel, and PowerPivot workbooks are Excel workbooks, you can leverage all the capabilities of Excel itself to do your analysis and reporting. Features such as PivotTables, PivotCharts, named sets, conditional formatting, and others are all available as you work with PowerPivot data.

Once you've built your PowerPivot application and you're happy with the capabilities it gives you to analyze your data and present it in an insightful way, you probably will want to share it with others. You now move into the realm of the other major component of PowerPivot — PowerPivot for SharePoint.

PowerPivot for SharePoint

PowerPivot for SharePoint installs on top of SharePoint 2010, and adds services and functionality to SharePoint to make PowerPivot workbooks first-class citizens of the SharePoint system. If you think of PowerPivot for Excel as the tool for creating and editing PowerPivot applications, PowerPivot for SharePoint is the tool that allows collaboration, sharing, and reporting. In addition, in conjunction with Excel Services, PowerPivot for SharePoint provides a "thin-client" capability for PowerPivot applications.

Because of the nature of SharePoint itself, installing PowerPivot for SharePoint is significantly more complex than installing PowerPivot for Excel. Hence, this introductory chapter won't spend time going into the details of PowerPivot for SharePoint installation. Those details will be presented in Chapter 8. For now, let's take a high-level look at the parts of PowerPivot for SharePoint so that you can get an idea of what it looks like and what it does.

PowerPivot for SharePoint consists of two main components that give SharePoint the capability of hosting PowerPivot applications. These two components are the *PowerPivot System Service* (the *PowerPivot Service* for short) and the *Analysis Services Service in VertiPaq mode* (or the *Analysis Services Service*). PowerPivot for SharePoint also includes a Web Service component that allows applications to connect to PowerPivot workbook data from outside the farm.

In the following discussions, remember that references to the Analysis Services Service are actually references to Analysis Services in VertiPaq mode.

Figure 1-6 shows an overview of the components that make up the PowerPivot for SharePoint system.

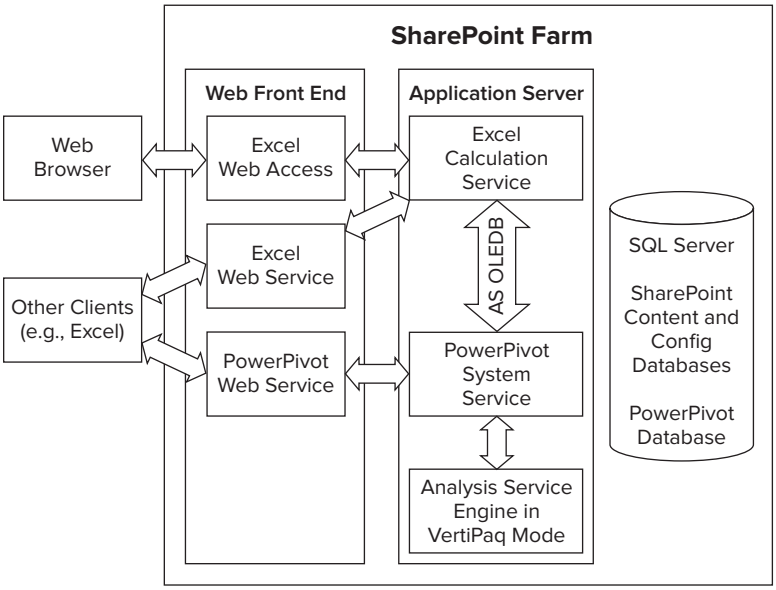

FIGURE 1-6: The components of PowerPivot for SharePoint

On the backend, PowerPivot for SharePoint includes one or more instances of the Analysis Services Service. These service instances are analogous to the traditional Analysis Services servers that you may be familiar with from previous versions of Analysis Services. Unlike those previous versions of Analysis Services, however, these services are under the control of the PowerPivot System Service. The PowerPivot System Service mediates the requests to load and work with the data in PowerPivot workbooks.

PowerPivot for SharePoint will load balance those requests among a pool of one or more Analysis Services Servers. If more requests come in than there are servers that can handle them, PowerPivot for SharePoint will unload the oldest unused workbooks in favor of the newest requests. You'll learn more about the VertiPaq mode of the Analysis Services engine later in this chapter.

When PowerPivot for SharePoint loads a workbook, it opens the workbook file (which, you will remember, contains the PowerPivot data store), extracts the PowerPivot data from the file, and sends it to an Analysis Services server that loads it as an in-memory analytical database. It also hooks up that server instance as an external data source to the corresponding published workbook. PowerPivot for SharePoint is smart about this. If more than one user requested a read-only copy of the same workbook, only one copy will be loaded, and the multiple workbooks will get connected to that single instance. This is just one of the ways that PowerPivot for SharePoint provides scalability for your self-service BI applications.

Note also, as shown in Figure 1-6, PowerPivot will create a database on the SharePoint farm's SQL Server to store various state information that must be queried from the different instances of the PowerPivot services.

PowerPivot for SharePoint also provides automatic data refresh from the source data of the workbook. Workbook owners can schedule this data refresh to happen at a periodic interval, or invoke a one-time refresh. When the time arrives to refresh the workbook, PowerPivot will load the data and tell the back-end server to reprocess it with the latest version of the source data. Once the data refresh is done, the workbook will be saved so that the next time a user asks for the workbook, it will contain up-to-date data. Figure 1-7 shows the Web page that allows scheduling of automatic data refresh.

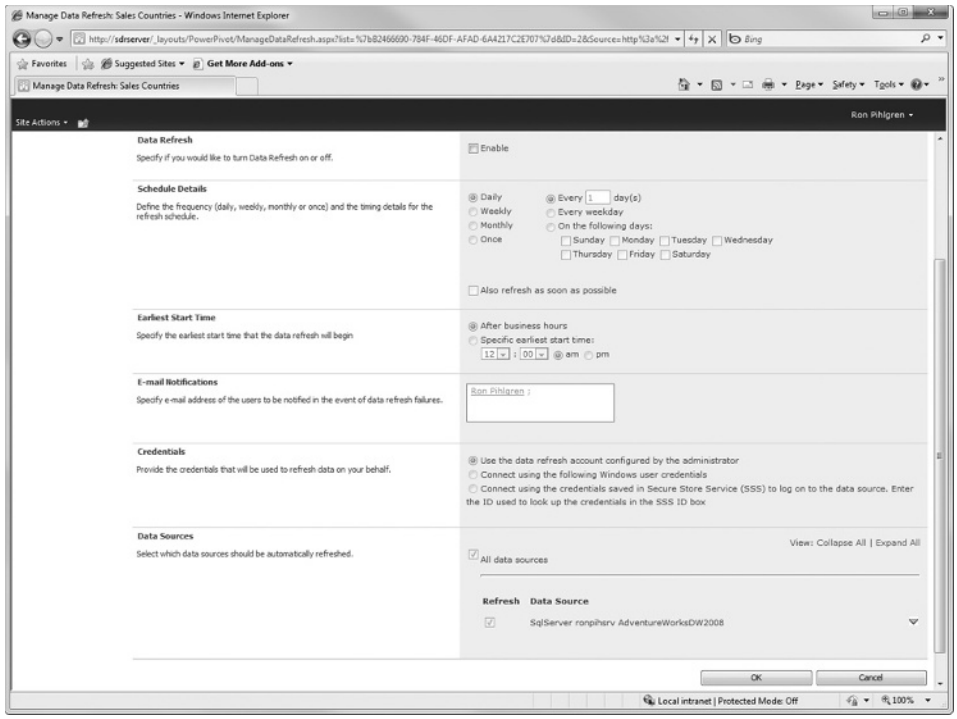

FIGURE 1-7: PowerPivot for SharePoint data refresh schedule page

As you may have guessed by now, PowerPivot for SharePoint has a lot of moving parts. A goal of the PowerPivot team was to make the management of the server side of PowerPivot easy. To do that, PowerPivot for SharePoint does extensive logging to SharePoint's Unified Logging Services (ULS) log. Becoming proficient at understanding ULS logs will give you a leg up on successfully managing and administering PowerPivot for SharePoint. Chapter 9 provides more detail on this.

The management dashboard is another feature of PowerPivot for SharePoint that helps in application management. The example shown in Figure 1-8 is a PowerPivot workbook that imports data from PowerPivot's usage database. It provides a helpful dashboard that gives the SharePoint administrators responsible for the server side of PowerPivot the capability to understand usage patterns of the PowerPivot applications that live in their SharePoint farm and to take appropriate action. High usage of a particular application, for example, may indicate that it is ready to be built out into a full-blown corporate BI application.

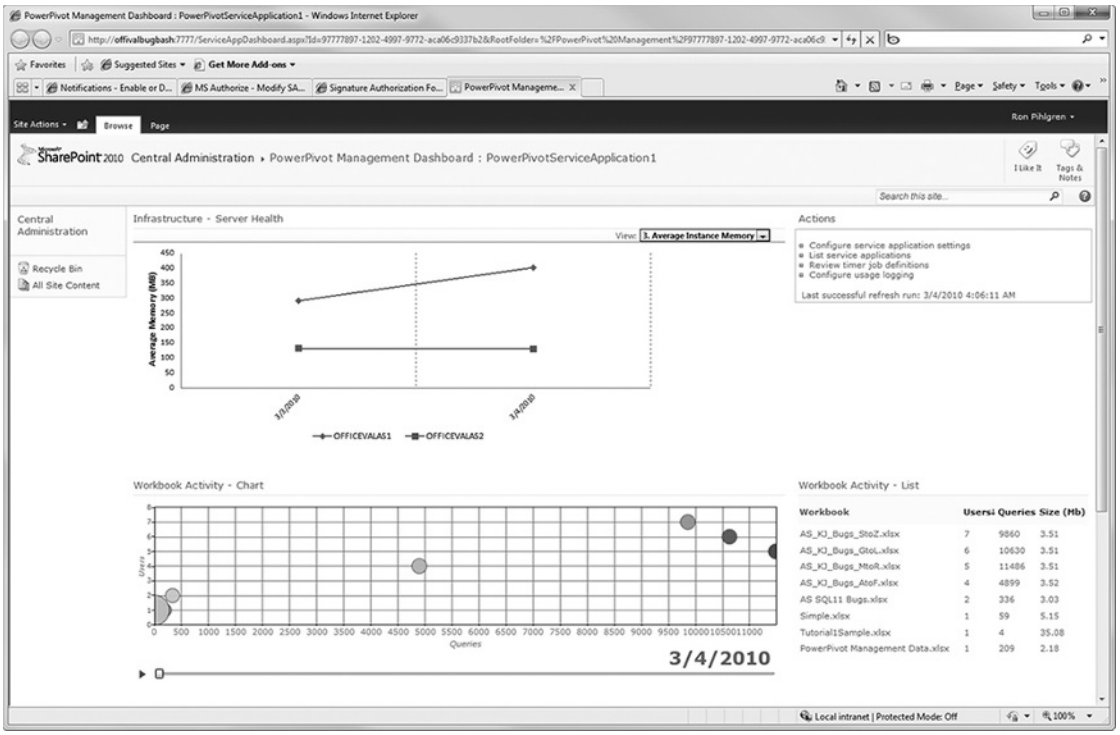

FIGURE 1-8: PowerPivot for SharePoint Management dashboard

Another feature of PowerPivot for SharePoint is the addition to SharePoint of a new type of document library designed to make working with PowerPivot applications pleasant. This includes the capability to browse that document library using rich views of the PowerPivot applications contained within them. This feature is called the *PowerPivot Gallery*. Figure 1-9 shows the default display of PowerPivot applications within the PowerPivot Gallery. Other views are also available. Figure 1-10 shows the carousel view of that same gallery.

CORPORATE BI AFTER POWERPIVOT

What happens to corporate business intelligence (corporate BI) after PowerPivot? The short answer is, "not much will change." PowerPivot was not designed to replace corporate BI. It was designed to add to it and bring the power of BI to people who don't have it today. It augments rather than replaces previous BI tools and techniques.

Analysis Services, SQL Server Management Studio, and BI Development Studio don't go away with the introduction of PowerPivot. Although the focus in this R2 release of SQL Server 2008 is on PowerPivot and self-service BI, future releases of SQL Server will add features that increase the usefulness and capability of the corporate BI side of the house.

There are capabilities corporate BI has that PowerPivot doesn't. For example, certain types of dimensions (such as parent/child dimensions) and certain features (such as writeback) are not available in PowerPivot applications. Furthermore, because of limitations in version one of PowerPivot, Workbooks cannot exceed 2 GB in size. Corporate BI databases in Analysis Services can be multiple terabytes in size.

It's important to realize these limitations of PowerPivot so that you don't try to use it in a way that it wasn't intended to be used and won't work. Understand the capabilities of each of the tools, and pick the right tool for the job.

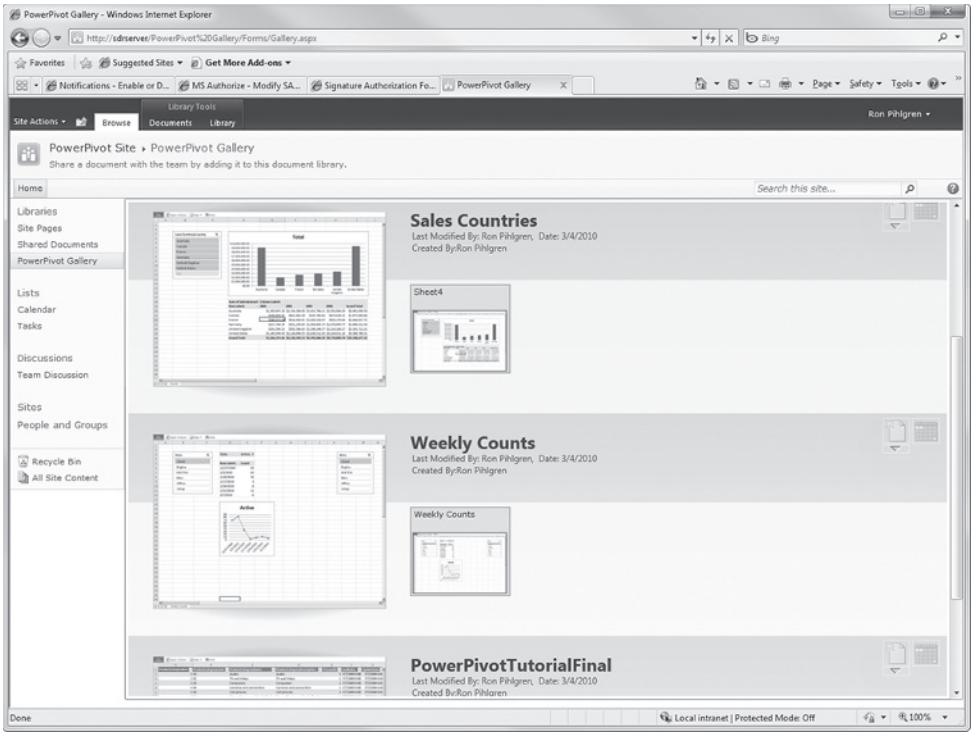

FIGURE 1-9: The PowerPivot Gallery (default view)

FIGURE 1-10: The PowerPivot Gallery (carousel view)

Being able to view your library of PowerPivot applications using these rich views is a lot more inviting and helpful than looking at a simple list of application filenames.

The Analysis Services Engine in VertiPaq Mode

A common component of both PowerPivot for Excel and PowerPivot for SharePoint is the *VertiPaq in-memory engine*. VertiPaq is actually a new mode of the Analysis Services server engine and has a number of innovative new features. It should be noted that, in a similar way that self-service BI is an extension (not a replacement) of corporate BI (see the sidebar, "Corporate BI After PowerPivot"), the Analysis Services engine in VertiPaq mode is an extension of (rather than a replacement for) the previous versions of the SQL Server Analysis Services server. The Analysis Services engine is still (as in the past) made up of two main components: the calculation engine and the storage engine.

The *calculation engine* is responsible for all calculations, including those that involve the new *data analysis expressions* (DAX) language (described later in this section). The calculation engine also supports Analysis Service's previous query language, *Multidimensional Expressions* (*MDX*). This

allows traditional Analysis Services client applications (such as Excel) to work without modification against PowerPivot data. A third query language that the calculation engine supports is tabular queries, which it uses to talk to relational databases.

The *storage engine* component of the Analysis Services engine is responsible for the importing of data from external data sources, as well as the efficient storage and retrieval of data in response to requests from the calculation engine.

The key difference between the traditional mode of the Analysis Services server (called the *Multidimensional Online Analytical Processing,* or *MOLAP*, mode) and the new VertiPaq mode is where the data is stored when it is operated on.

In MOLAP mode, the data is stored on disk. This is the traditional mode of most typical database engines. It made sense that this was the only mode of storage when RAM was expensive and disk storage was relatively cheap. Relying on the normal amount of RAM in a typical system to hold all the data in large databases wasn't feasible in those days, especially for analytical data, where analysis is usually based on calculations against very large data sets.

In today's world, this is starting to not be the case. RAM in large quantities has become very cheap relative to where it was even just a few years ago. Now, having enough RAM to store large quantities of data in typical machines is much more feasible. When you can consider doing this, different capabilities become available. First, accessing data from RAM is much faster (orders of magnitude faster) than accessing data from disk. This allows higher performance when working with data than was possible before.

In the BI world, performance is king. Being able to interact with your data (that is, slicing and dicing in real time as ideas flow and questions about the data form) is a big part of the power of BI. Because of this, disk-based BI servers would do things like pre-aggregate data in anticipation of the way analysts would want to slice and dice it. If calculations could be done fast enough, though, having to do all that pre-aggregation with its attendant processing and storage requirements would not be necessary. This is one of the big advantages of an engine that works with data in-memory.

Although RAM has become much cheaper and more plentiful, it is still more costly than disk storage. It's not totally free! And there is a limit on the amount of RAM that systems can have. Clearly, you still must be cognizant of the amount of RAM you need and optimize your usage of it. The VertiPaq engine still must make the most of the RAM that is available.

Because of this, the VertiPaq mode also includes patented compression technology that is made possible by implementing what is called a *column-oriented store*. This means that, unlike traditional database storage engines that store data by rows, Analysis Services in VertiPaq mode stores data by columns. This allows much more effective compression of tabular data. VertiPaq compression rates depend on the nature of the data stored, but rates greater than 10:1 are common.

Another goal of self-service BI is to provide the capability to make building analytical applications much simpler than building analytical applications using corporate BI tools. With that in mind, the VertiPaq engine implements the new DAX expression language that allows self-service BI practitioners to create calculations in a manner very close to working with Excel formulas.

There are a couple of things to keep in mind about DAX.

First, PowerPivot is in its first version. In order to ship the first version in a reasonable timeframe, not all the functionality that the team would have liked to include is implemented in this first version of the product. The team believes that it has made the right trade-offs in terms of useful functionality and shipping this version of the product. You can expect DAX to grow in functionality in future versions of PowerPivot.

Second, although DAX expressions are designed to be similar to Excel formulas, they are different in a fundamental way. DAX is not designed to operate on single cells of data. It is designed to work on groups of data (such as columns) simultaneously. This is key to the way that BI analysis is done. So, when you write a DAX expression for a calculated column of a table, that expression will apply to that entire column for every row in the table. You'll learn much more about DAX in later chapters of this book.

INSIDE POWERPIVOT

PowerPivot was born from two 2006 Think Week papers by Amir Netz, who was at that time the Microsoft Business Intelligence partner-level architect. (As of the writing of this book, Netz has been made a Microsoft Distinguished Engineer.)

Think Weeks were dedicated periods of time that Bill Gates, when he was head of Microsoft, would take to immerse himself in reading and thinking about new trends and concepts that could impact the future direction of Microsoft. Microsoft employees were encouraged to write papers that would be a part of Gates' Think Week reading. Compelling Think Week papers could lead to allocation of resources to make new products and technologies happen. In the case of the papers that led to PowerPivot, this was the case.

The first paper was about the concept of a BI "sandbox"—a product that would allow much easier BI application creation in a defined and controlled space that would include a relational database, a multidimensional database, and a report generator. Although some of what was in that paper changed as PowerPivot grew (for example, the initial thought was that Microsoft Access would be the PowerPivot client), many of the ideas remain, and are core to what PowerPivot is today.

The second Think Week paper proposed an in-memory BI engine that took advantage of the emerging trends in computer hardware (such as falling RAM prices and multi-core processors) that would allow such an in-memory engine to be both possible and practical. The capabilities that such an engine would make possible included some of the features in the previous paper.

Both papers were well-received and a small incubation team was formed to explore the possibility of building a product incorporating the ideas in the papers. At the time, the project was called the "Sandbox Project" after the title of the first of the Think Week papers. This incubation team spent the second half of the SQL Server 2008 product cycle putting together specifications, designs, and plans under the code name "Gemini." Toward the end of the 2008 product cycle, the go-ahead was given to turn the project into a product, and the rest is history.

SUMMARY

This chapter examined the attributes of self-service BI and Microsoft's vision for self-service BI. It discussed Microsoft's first version of its self-service BI product, PowerPivot. It described the components that make up PowerPivot, and discussed the features of each of those components. The goal of this introductory chapter was to whet your appetite to learn more about this great new capability of self-service BI being introduced in Microsoft SQL Server 2008 R2.

Chapter 2 walks you through an example of using PowerPivot from end to end. This will give you a solid grounding in the entire PowerPivot product and prepare you for digging into greater detail in subsequent chapters.